External Interface Server Guidance in Pro:Centric Direct v4.0

Document Number: PCN-CI-4.0

Author: Myung Il Seo

Date: September 10, 2019

CONTENTS

1 Revision History

Date	Issue	Reason for change	Changed by
April 2nd 2017	Draft 1	New Document	Myung Il Seo
May 23, 2018	Draft 2	Modified sample json	Myung Il Seo
July 23, 2018	Draft 3	Modified message json from PMS & Adding PMS notification events for QMS	Myung Il Seo
July 28, 2018	Draft 4	Adding sample JSON to notify events for /requests QMS API	Myung Il Seo
October 30, 2018	Draft 4	Adding categories function described for PCD 3.0	Myung Il Seo
April 23, 2019		Modification for PCD 3.5 (QMS v3, IoT)	Myung Il Seo
September 10, 2019		Adding TV control feature	Myung Il Seo
January 23, 2020		Adding PMS status events	Myung Il Seo

2 External Interface Server Client Guide

This section is a guide for External Interface Server to connect and utilize the Pro:Centric Server(PCS) system. The request handler function is designed to consolidate requests for various system data and forward these requests to the appropriate subsyetm for processing. Requests can come from the Admim client function or from the television or potentially in the future from 3rd party sub systems or interfaces. The results of the requests are forwarded to the distribution manager to typically be sent to the the television. An exemplar use case would be the guest requesting their billing information. The television would request the folio summary and the request handler would call the data handler which would access the information from the property management system and forward the response to the distribution manager which would provide it back to the television.

External Interface Server could be located outside of PCS or inside PCS by using docker(http://www.docker.com/) platform.

3 Overview

PCS Server requests to External Interface Server for getting data from 3rd party External Servers such as PMS(Property Management System) or QMS(Quality Management System).

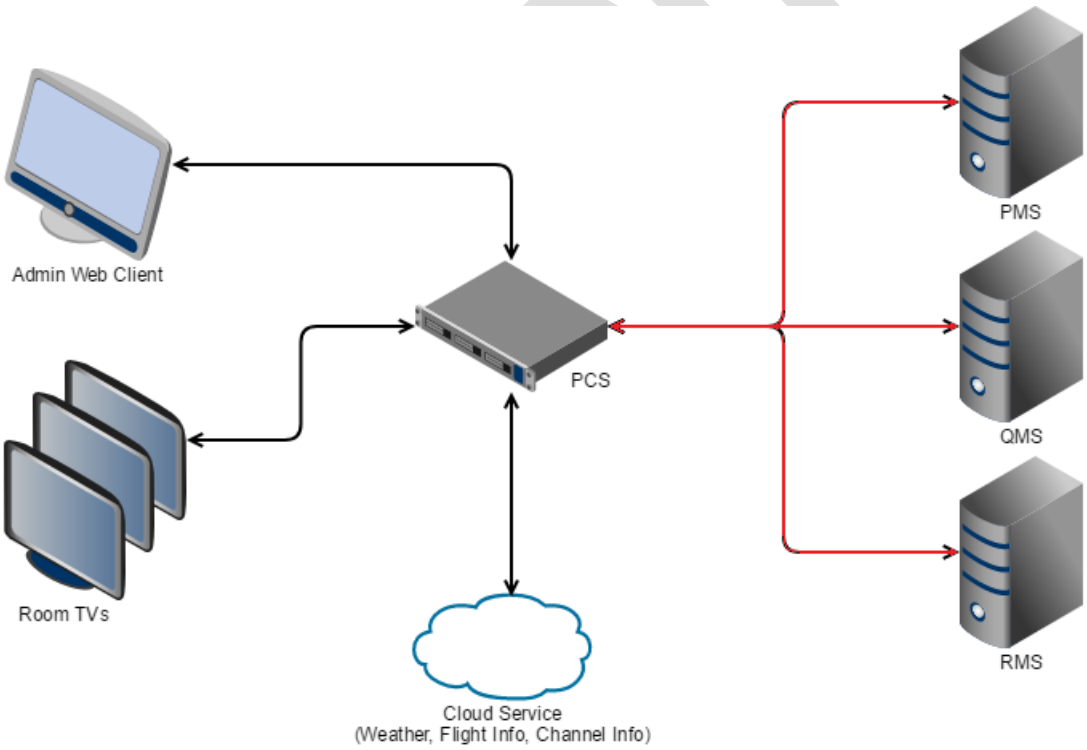

Figure 1

The Pro:Centric Server (PCS) will query the Interface Server as required for information and updates. The syntax of the interchange is outlined below in the command set. The Licensee is responsible for implementing the interface between the PCS and each Interface Server

3.1 General Concepts

The PMIS must implement an HTTP server to support the RESTful command structure outlined below. A PMIS server defines the service endpoint URL through which PCS clients may access and interact with hospitality related resources such as a property management system (PMS), ticket management systems (TMS), or point of sale system (POS). The PMIS outputs these resources using the JavaScript Object Notation (JSON).

3.2 Documentation Conventions

- Example resource representations, elements of representation, and computer code or commands are shown in a `monospace` font.
- URLs shown in examples are informational only. The URLs shown are not intended to define a required URL format or structure.

3.3 Compliance

An implementation is not compliant if it fails to satisfy one or more of the MUST or REQUIRED level requirements. An implementation that satisfies all the MUST or REQUIRED and all of the SHOULD level requirements is said to be "unconditionally compliant." One that satisfies all of the MUST level requirements but not all the SHOULD level requirements is said to be "conditionally compliant."

The following key words are to be interpreted as described in RFC 2119: MUST, MUST NOT, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL.

3.4 Constraints

- All resource representations provided by the PMIS MUST be valid JSON documents.
- A PMIS may provide templated URLs formatted according to RFC 6570
- The service endpoint of a compliant implementation MUST provide a site's representation.
- Servers MAY require clients to support HTTP Authentication (BASIC, DIGEST, or other) for some requests.
- Servers MAY provide additional relations, semantics, and data descriptions (objects, properties, elements, attributes, etc.) not covered in this profile, but these additions MUST NOT contradict the semantics outlined here.

4 External Server Setting & Authentication

4.1 External Server Setting

An admin should input the external server information at PCS's Admin Web Client. The menu is Setting → External Server. According to the server name, the input areas are changed. A detailed description of the external server settings can be found in the Check API for each server.

Explanation about the input area:

- Sever Name : choose the external server type. (PMS, QMS, Pay Channel, Room Control, ES)
- Use Button : Select ON or OFF for the server connection. If selection is On, PCS checks the connection to the external server.
- Server Type : Choose Server type. Usually the type is open API. But some hotel chains want to put their interface modules in PCS, so it could be company's name or hotel name. If Server name is QMS, this element is changed to QMS Type.
- Auto power on : The TV in the hotel room for the guest is auto power on and shows welcome message when PCS receives Check-in event from External Interface Server. And enable time range can be set.

- Type : Using SSL or not. It depends on external server security.
- Host :The External Interface Server IP address.
- Port : The External Interface Server port number.
- User : Additional Security method to connect the External Interface Server for User ID.
- Password : Additional Security method to connect the External Interface Server for User password.
- Polling Time : Choose Event driven by subscription or Polling by input the polling seconds. Refer the Subscription section. In case of Open API, polling to get events is not supported currently. If the pollingSeconds value is 0, the event is received by subscription events.
- Authorization Section : This section is for authentication or to distinguish hotels. See the next authentication section.

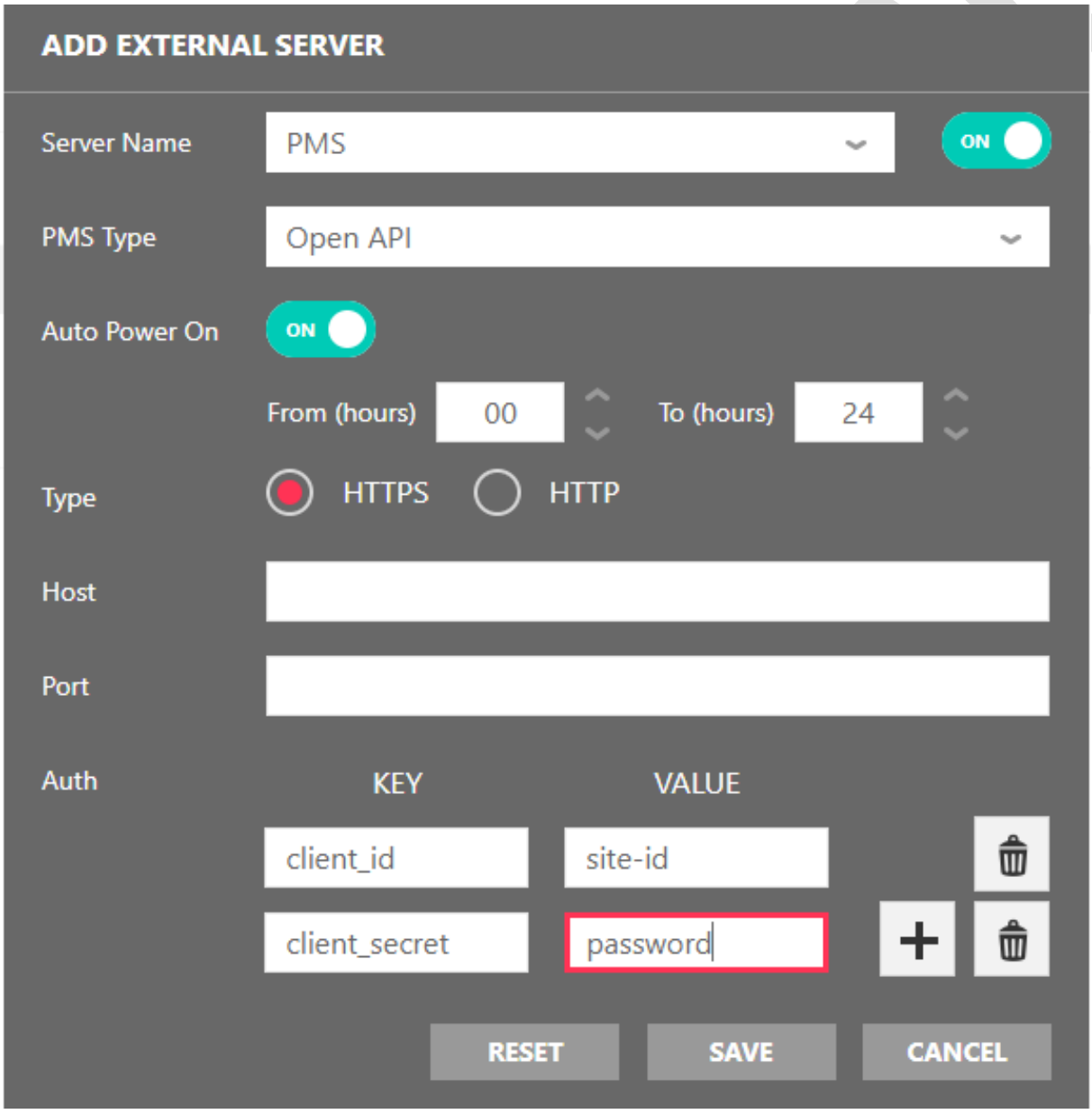

Figure 2

4.2 Authentication

PCS supports two kinds of authenticaton for External Interface Server. The first thing is by putting User ID and Password at HTTP(S) Site information. Another is by input the custom field at Authorization section. The input values in Auth section are sent when PCS checks the connection to the external server. The auth values are included in Head of every requests from PCS.

- Click + button and input Key and Value fields.

If the external server want to call PCD's open API, it has to put token value in the header of calling API. The token value can be obtained by using client_id and client_secret as parameters for the invocation authentication API.

4.2.1 Getting access token

To call PCD's open API, the external server uses token in the header of calling API. The token value can be obtained by using client_id and client_secret as parameters for the invocation authentication API. Also the default port of PCS is 60080. For example, you can call "https://192.168.0.11:60080/api/v2/auth/tokens" api to get token values.

4.2.1.1 POST /auth/tokens

Request an access token using the client_id and client_secret entered in the external server settings page of the administrator client.

4.2.1.1.1 Request : POST /api/v2/auth/tokens

1. Headers
   1.1 Content-Type: application/json
2. Body
   2.1 client_id : (mandatory) usually hotel ID getting from PMS
   2.2 client_secret : (mandatory) unique indivisual value likes password.
   2.3 grant_type : (mandatory) input "client_credentials" value

4.2.1.1.2 Example

Body

```
{
    "client_id": "0123456",
    "client_secret": "secret1",
    "grant_type": "client_credentials"
}
```

4.2.1.1.3 Responses

1 200 (OK)
   1.1 Headers
      1.1.1 Content-Type: application/json
   1.2 Body
      1.2.1 access_token : new token
      1.2.2 refresh_token : using refresh_token when access_token is expires
      1.2.3 expires_in : expired duration, seconds (Default expired period is 1 week)
      1.2.4 token_type : token value's title
2 401 (Unauthorized)
3 404 (Not Found)

4.2.1.1.4 Response Example


```
{
 {
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.IjJ1UkZ2a2hheXdlWDVReTRvTWZlNmFyR2pQVmVqbURLQ",
  "refresh_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.IjlWWEpJM01QVVdOOE5TdG10SkJSZDhoWm82SXdMNWh",
  "expires_in": 604800,
  "token_type": "Bearer"
 }
}
```


4.2.2 Refresh access token

After a week to receive access token, it will need to get a new access token. At that time, the external server should use the refresh_token value.

4.2.2.1 POST /auth/tokens

Request an access token using the client_id and client_secret entered in the external server settings page of the administrator client.

4.2.2.1.1 Request : POST /api/v2/auth/tokens

1. Headers
   - 1.1 Content-Type: application/json
2. Body
   - 2.1 client_id : (mandatory) usually hotel ID getting from PMS
   - 2.2 client_secret : (mandatory) unique indivisual value likes password.
   - 2.3 grant_type : (mandatory) input “client_credentials” value
   - 2.4 refresh_token : (mandatory) the value when the access_token was previously obtained.

4.2.2.1.2 Example

Body


```
{
        "client_id": "0123456",
        "client_secret": "secret1",
        "grant_type": "client_credentials",
        “refresh_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.IjlWWEpJM01QVVdOOE5TdG10SkJSZDhoWm82SXd"
}
```


4.2.2.1.3 Responses

1 200 (OK)
   - 1.1 Headers
     - 1.1.1 Content-Type: application/json
   - 1.2 Body
     - 1.2.1 access_token : new token
     - 1.2.2 refresh_token : using refresh_token when access_token is expires
     - 1.2.3 expires_in : expired duration, seconds
     - 1.2.4 token_type : token value’s title

2 401 (Unauthorized)

3 404 (Not Found)

4.2.2.1.4 Response Example


```
{
```

```
 {
   "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.IjJ1UkZ2a2hheXdlWDVReTRvTWZlNmFyR2pQVmVqbURLQ",
   "refresh_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.IjlWWEpJM01QVVdOOE5TdG10SkJSZDhoWm82SXdMNWh",
   "expires_in": 604800,
   "token_type": "Bearer"
 }
}
```

4.2.3 Usage access token

All requests are requested by putting the access token value in the HTTP header.

- Request Example

```
GET /api/hoteltv/v2/concierges HTTP/1.1
Host: localhost:60080
Authorization: Bearer <Access Token>
Cache-Control: no-cache
```

- Response Example

```
Content-Type: application/json; charset=utf-8
{
 "apiVersion": "2.0",
 "data": {
   ...
 }
}
```

4.3 Check the connection

After an admin input the external server information, PCS checks the connection with External interface server when the isUse value is true. At that time PCS uses the user ID, password and Authentication values for security. Refer the Figure 3. If the external server want to call PCD's open API, it has to put token value in the header of calling API. The token value can be obtained by using client_id and client_secret as parameters for the invocation authentication API.

- Request Example

```
POST /api/pms/v2/check HTTP/1.1
Host: 10.20.30.40:8880
Content-Type: application/x-www-form-urlencoded
Cache-Control: no-cache

{
 "host": "10.20.30.40",
 "port": 8880,
 "user": "lge",
 "password": "123456",
 "auth": {
   "client_id": "123456789",
   "client_secret": "lge"
 }
}
```


- Response Example

200 "OK"

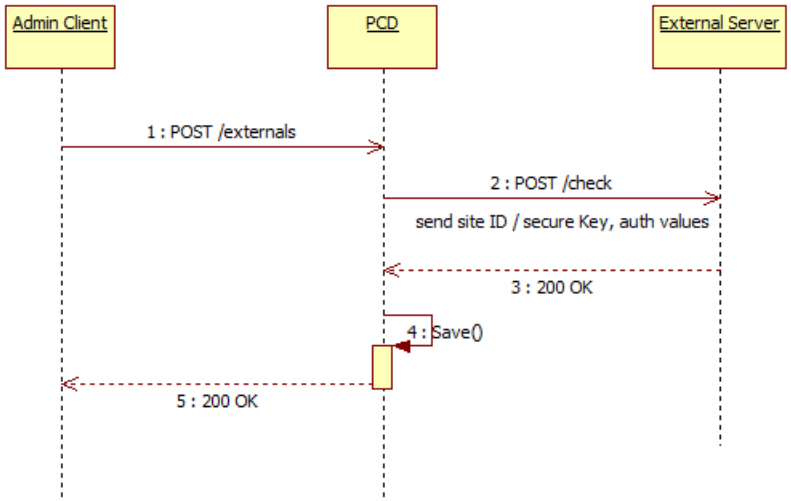

Figure 3

4.4 Initiation

If an admin clicks “Fetch Settings” button at the external server setting page in Admin Client, PCS initializes to synchrounous data with external server. PCD calls below APIs to external server. The PMS interface server has to provide below APIs.

4.4.1 PMS initiation

- GET /details : getting sites information from PMS. Refer the 5.2.4.1 section.
- GET /rooms/{roomId} : getting rooms statues from PMS. This API calls the number of rooms the PCD has. Refer the 5.2.5.1 section.
- POST /subscriptions : registering PMS interface server to get events from PMS. Refer the 5.2.9.1 section.

Refer the Figure 4.

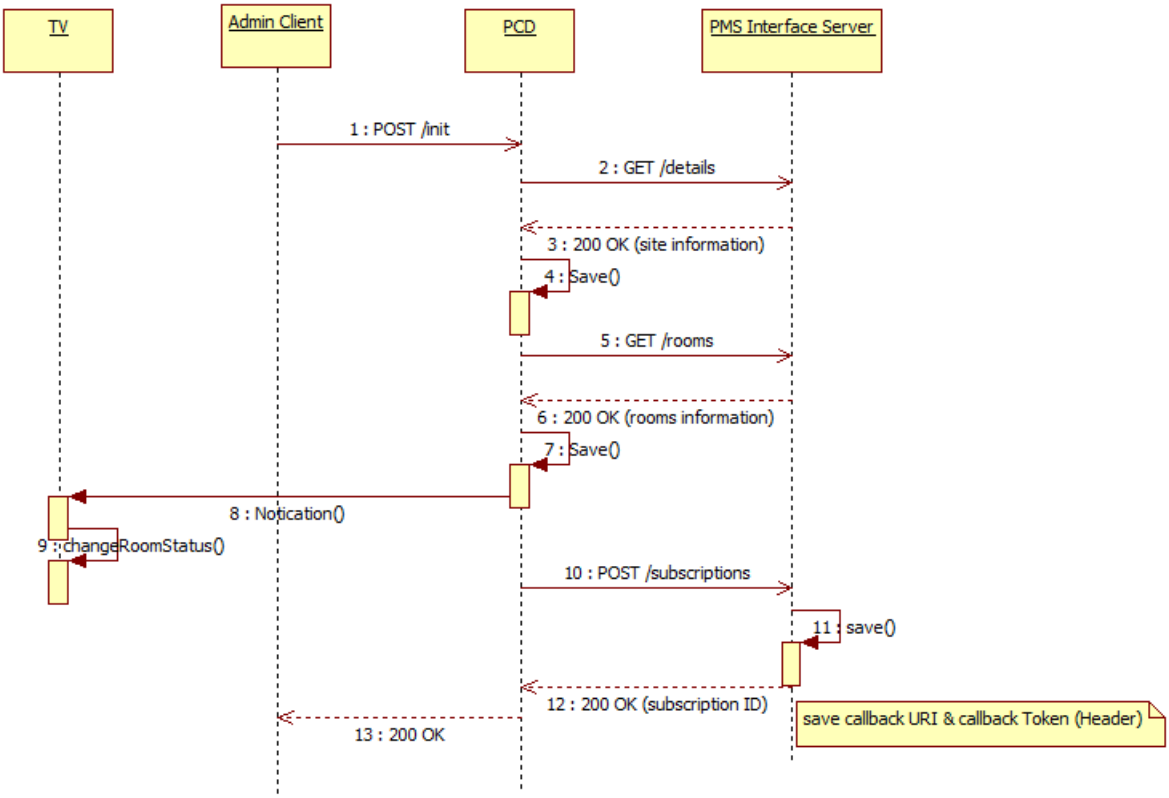

Figure 4

4.4.2 QMS initiation

- GET /directories : getting sites information(Restaurant information, Facility information, etc) from QMS
- GET /services : getting services kinds (towel, make up room, etc) from QMS
- POST /subscriptions : registering QMS interface server to get events from QMS

Refer the Figure 5.

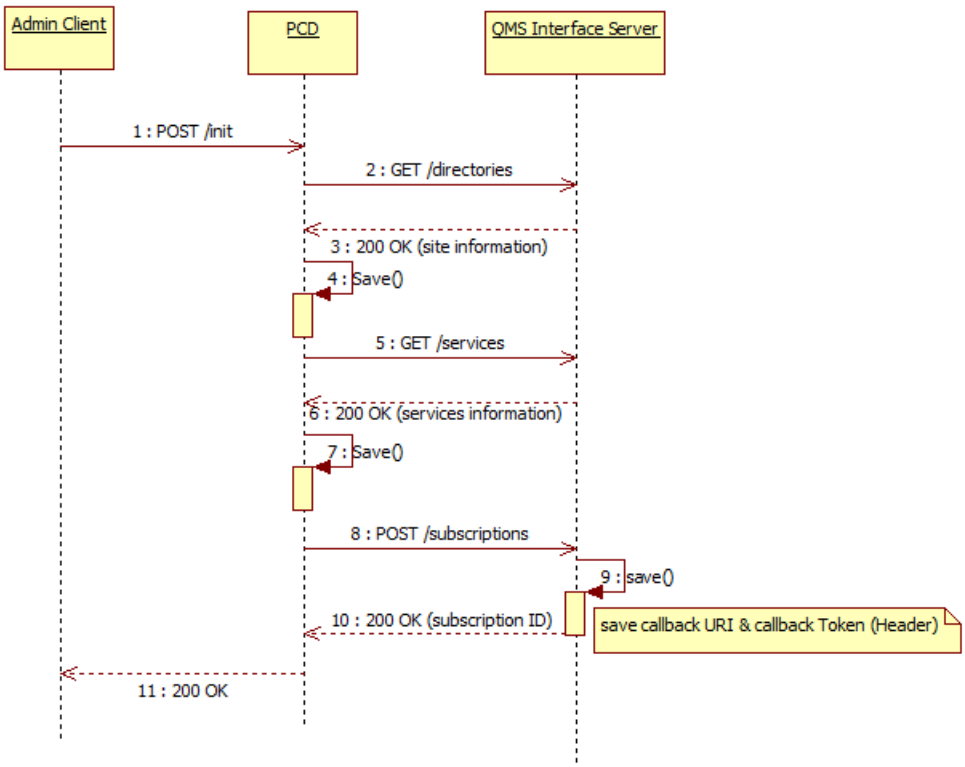

Figure 5

4.4.3 Pay Channel initiation

- POST /subscriptions : registering pay channel interface server to get events. Refer the 5.4.8.1 section.
- POST /sites/{site_id}/channels : synchronizing with pay channel server by getting channel information. Refer the 5.4.5.1 section.

5 API Client

5.1 Semantic Profile

What follows is a list of link relation definitions and their associated semantics. Servers SHOULD respond to requests with resource representations that contain links and/or embedded resources identified by the relations below and resource state data. Link relations are defined in generalized terms to allow for the possibility that the relations may be used in a variety of contexts.

- Clients MUST be able to process the JSON type and SHOULD be prepared to handle all of the JSON properties, objects, and arrays described below. Clients should also be prepared to provide state transfers by the submission of valid representations to the server as indicated. Servers MAY provide additional semantics and clients MAY support those additions through private agreements.

- In all resource representations, a JSON property without a value MUST be null (`{ "prop":null }`). An empty string ("") or 0 value SHALL NOT be used to represent an unknown property value. A JSON array without values MUST be an empty array (`{ "arr":[] }`) and a JSON object without members MUST be an empty object (`{"obj":{}}`).

- Clients MAY use HTTP POST to create selected resources and PUT to update selected resources. When POSTing or PUTing a resource representation, the client WILL use a content type of "application/json" and WILL NOT include any HAL markup in the representation. The client MAY not post URI link targets. The URI space is controlled by the server.

- Clients SHOULD treat all URIs provided by the server as opaque identifiers and/or locators. A client WILL NOT bookmark URIs other than the service endpoint or compose server URIs unless guided by a URI template.

- Cache-control headers defined in the HTTP Methods section of each relation definition are advisory only. A server MAY choose different cache directives.

5.2 API Client – PMS

5.2.1 Prefix

It is defined prefix to call PMS APIs.

/api/pms/v2

5.2.2 Check

Check API verifies PMS server information and registers it in PMS Interface server. PCD calls this API when an admin clicks the “SAVE” button after putting the server information in the Admin Client.

5.2.2.1 POST /check

POST /check API verifies PMS server information passed through request body and register it only if it is a valid information. In case of Open API, polling to get events is not supported currently. If the pollingSeconds value is 0, the event is received by subscription events.

5.2.2.1.1 Request : POST /api/pms/v2/check

1. Headers

1.1 Content-Type: application/json
1.2 auth
1.2.1 client_id
1.2.2 client_secret

2. Body
2.1 name
2.2 host
2.3 port
2.4 isSSL
2.5 auth
2.5.1 client_id
2.5.2 client_secret
2.6 pollingSeconds

5.2.2.1.2 Example

Headers

```
{
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        }
}
```

Body

```
{
        "name": "PMS",
        "pmsType": "OPENAPI",
        "host": "10.177.223.41",
        "port": 8880,
        "isSSL": 0,
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        },
        "pollingSeconds": 0
}
```

5.2.2.1.3 Responses

1 200 (OK)
1.1 Headers
1.1.1 Content-Type: application/json
1.2 Body
1.2.1 status
2 401 (Unauthorized)
3 404 (Not Found)

5.2.2.1.4 Response Example

```
{
        "status": "success"
}
```

5.2.3 Statuses

Status API checks current status of PMS server

Attribute	Type	Description
id	Integer	Unique identifier for status event.
created	String	Time when the status information was generated. ISO8601 standard format.
status	String	PMS server status. (up | down)

5.2.3.1 GET /statuses

Get /statuses API returns current status of PMS Server. E.g. up, down, etc.

5.2.3.1.1 Request : GET /api/pms/v2/statuses

1. Headers
   1.1 auth
      1.1.1 client_id
      1.1.2 client_secret

5.2.3.1.2 Example

```
Headers
{
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        }
}
```

5.2.3.1.3 Responses

1 200 (OK)
   1.1 Headers
      1.1.1 Content-Type: application/json
   1.2 Body
      1.2.1 id
      1.2.2 created
      1.2.3 status
2 401 (Unauthorized)
3 404 (Not Found)

5.2.3.1.4 Example

```
{
        “status”: “success”,
        “data”: {
                "id": 952,
                "created": "2017-05-27T19:11:20.203Z",
                "status": "up"
        }
}
```

5.2.4 Sites

Refers to a single property such as a hotel at which hospitality system services are available

Attribute	Type	Description
id	String	globally unique site identifier. It is same as client_id in the auth section.
name	String	name of the site

currency	String	Currency of the site. It will be displayed at folio view.
website	String	Website of the site
timezone	String	Timezone abbreviation (ex. CDT, KST)
contacts	Object	
telephone	String	Admin's telephone number of the site
email	String	Admin's Email address of the site
location	Object	
address	Object	
street	String	Street name of the site address
city	String	City name of the site address
state	String	State name of the site address
postcode	String	Postal code of the site address
country	String	Country name of the site
coordinate	Object	
latitude	String	Latitude of the site location
longitude	String	Longitude of the site location
timestamp	String	Current time of PMS server. ISO8601 standard format.

5.2.4.1 GET /details

Retrieve a site(hotel) reprentation. PCD calls this API at initiation time.

5.2.4.1.1 Request : GET /api/pms/v2/details

1. Headers
    1.1 auth
        1.1.1 client_id
        1.1.2 client_secret

5.2.4.1.2 Example

Headers
```
{
    "auth": {
            " client_id": "123456789",
            " client_secret": "lge"
    }
}
```

5.2.4.1.3 Responses

1 200 (OK)
    1.1 Headers
        1.1.1 Content-Type: application/json
    1.2 Body
        1.2.1 id
        1.2.2 name
        1.2.3 currency
        1.2.4 website
        1.2.5 timezone
        1.2.6 contacts
            1.2.6.1 telephone
            1.2.6.2 email

1.2.7 location
1.2.7.1 address
1.2.7.1.1 street
1.2.7.1.2 city
1.2.7.1.3 state
1.2.7.1.4 postcode
1.2.7.1.5 country
1.2.7.2 coordinate
1.2.7.2.1 latitude
1.2.7.2.2 longitude
1.2.8 timestamp
2 401 (Unauthorized)
3 404 (Not Found)

5.2.4.1.4 Example


```
{
	“status”: “success”,
	“data”: {
		"id": "0000006",
		"name": "Coyote",
		"currency": "USD",
		"website": "http://...",
		"timezone": "CDT",
		"contacts": {
			"telephone": "555-555-5555",
			"email": "hotel@example.org"
		},
		"location": {
			"address": {
				"street": "100 W State St",
				"city": "Chicago",
				"state": "IL",
				"postcode": "60304",
				"country": "USA"
			},
			"coordinate": {
				"latitude": "41.87150",
				"longitude": "-87.635420"
			}
		}
		"timestamp": "2017-05-27T19:11:20.203Z "
	}
}
```


5.2.5 Rooms

Room refers to a single guest room within a site.

Attribute	Type	Description
id	string	unique room number/identifier within a site
guests	array	Collection of guest object within the room. Refer to Guests

5.2.5.1 GET /rooms/{room_id}

Retrieve a room information.

5.2.5.1.1 Request : GET /api/pms/v2/rooms/{room_id}

1. Headers
   - 1.1 auth
     - 1.1.1 client_id
     - 1.1.2 client_secret

5.2.5.1.2 Example

```
Headers
{
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        }
}
```

5.2.5.1.3 Responses

1 200 (OK)
   - 1.1 Headers
     - 1.1.1 Content-Type: application/json
   - 1.2 Body
     - 1.2.1 id
     - 1.2.2 guests

2 401 (Unauthorized)

3 404 (PMS has no data) : Room not found. It means that the room is unoccupied.

5.2.5.1.4 Example

```
{
        "status": "success",
        "data: {
                "id": "101",
                "guests:" [
                        {
                                "name": {
                                        "prefix": null,
                                        "first": null,
                                        "middle": null,
                                        "last": null,
                                        "suffix": null,
                                        "full": "enda"
                                },
                                "balance": null,
                                "language": null,
                                "no_post": null,
                                "vip_status": null,
                                "id": "101-2",
                                "option": null,
                                "channel_preference": null,
                        },
```

```
                {
                        "name": {
                                "prefix": null,
                                "first": null,
                                "middle": null,
                                "last": null,
                                "suffix": null,
                                "full": "enda"
                        },
                        "balance": null,
                        "language": null,
                         "no_post": null,
                        "vip_status": null,
                        "id": "101-3",
                         "option": null,
                         "channel_preference": null,
                }
        ]
    }
}
```


5.2.6 Folios

Folio refers to an invoice detailing the charges, credits, and payments incurred or made by a room. A guest can see one's billing information when a guest want to checkout in one's room thru TV. Refer the Figure 6.

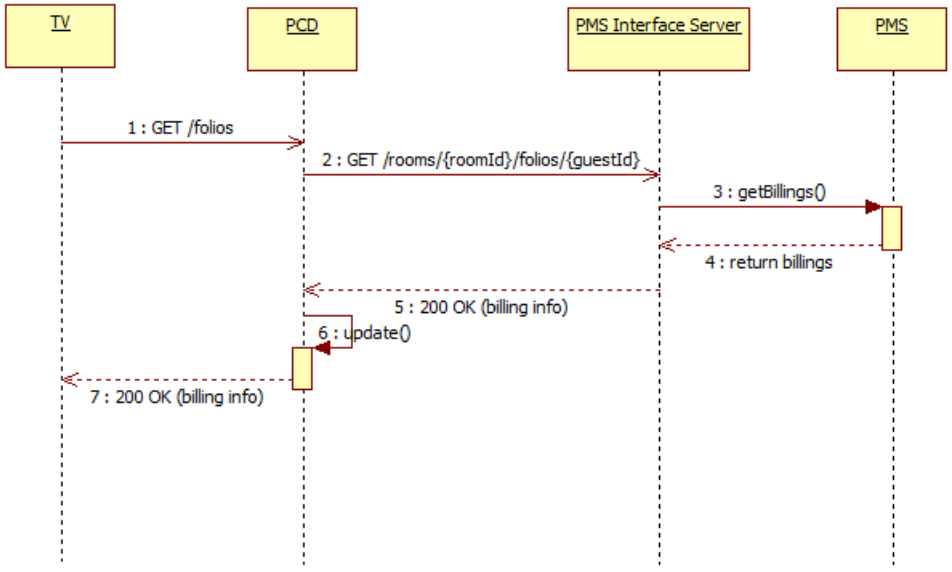

Figure 6

Attribute	Type	Description
id	string	unique among folios within the site
status	string	• `open` - items may be added to the folio

		• `locked` - the folio is temporarily unable to be updated. • `closing` - the folio is in the process of being closed. • `closed` - the guest associated with the folio is checked out.
balance	number	balance due on the folio
items	array	Collection of folio item. Folio item shows id, amount, create time, description, display properties. Folio item is displayed only if display property value of folio item is true. Refer to Folio Items

5.2.6.1 GET /rooms/{room_id}/folios/{guest_id}

Retrieve a full folio representation.

5.2.6.1.1 Request : GET /api/pms/v2/rooms/{room_id}/folios/{guest_id}

1. Headers
   1.1 auth
      1.1.1 client_id
      1.1.2 client_secret

5.2.6.1.2 Example

Headers
```
{
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        }
}
```

5.2.6.1.3 Responses

1 200 (OK)
   1.1 Headers
      1.1.1 Content-Type: application/json
   1.2 Body
      1.2.1 id
      1.2.2 status
      1.2.3 balance
      1.2.4 items – array
         1.2.4.1 id
         1.2.4.2 created
         1.2.4.3 description
         1.2.4.4 amount
         1.2.4.5 display
2 401 (Unauthorized)
3 404 (Not Found) Folio Not Found

5.2.6.1.4 Example

```
{
        "id" : "101-1",
        "status": "open",
        "balance": 65.19,
        "items": [
                {
```

```
                "id": 1,
                "created":"2012-05-25T17:27Z",
                "description":"in room movie",
                "amount":5.19,
                "display": true
            },
            {
                "id": 2,
                "created":"2012-05-25T17:27Z",
                "description":"room charge",
                "amount":60,
                "display": true
            }
        ]
}
```


5.2.7 Folio Items

Represents a folio item. A folio representation MAY contain zero or more folio item. A client may send a folio items on a server in the body of an HTTP POST request to post a charge or credit to a folio.

Attribute	Type	Description
amount	number	contains a signed floating-point numeric value indicating the amount of the debit or credit. Amont = subtotal + tax
description	string	contains TEXT data describing the debit or credit
purchase_id	string	It would be generated by client to identify the purchase. The PMS will NOT send this ID back when getting folio items.
revenue_code	number	This value needs to be configurable. The PMS vendor will provide the revenue_code to use.
subtotal	object	contains a signed floating-point numeric value indicating the amount of the debit or credit
subtotal.id	string	Unique id for subtotal
subtotal.amount	number	Subtotal amount
tax	object	contains a signed floating-point numeric value indicating the amount of Tax for the subtotal
tax.id	string	Unique id for subtotal
tax.amount	number	Subtotal amount

5.2.7.1 POST /rooms/{room_id}/folios/{guest_id}/folio-items

Request a charge from TV to PMS. Subtotal. ***The guest charge should always include the subtotal.***

5.2.7.1.1 request : POST /api/pms/v2/rooms/{room_id}/folios/{folio_id}/folio-items

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Content-Type: application/json
  - 1.1.2 auth
    - 1.1.2.1 client_id
    - 1.1.2.2 client_secret
- 1.2 Body
  - 1.2.1 amount
  - 1.2.2 description
  - 1.2.3 purchase_id

1.2.4 revenue_code
1.2.5 subtotal
1.2.5.1 id
1.2.5.2 amount
1.2.6 tax (OPTIONAL)
1.2.6.1 id
1.2.6.2 amount

5.2.7.1.2 Example

Headers
{
"auth": {
" client_id": "123456789",
" client_secret": "lge"
}
}
Body
{
"amount": 18.11,
"description": "TV Services",
"purchase_id": "ARAAnQAA",
"revenue_code": "1",
"subtotal": { "1": 17.99 },
"tax":{ "1": 0.12 }
}

5.2.7.1.3 Responses

1 200 (OK)
1.1 Headers
1.1.1 Content-Type: application/json
1.2 Body
1.2.1 status
1.2.2 data
2 401 (Unauthorized)
3 404 (Not Found)

5.2.7.1.4 Example

{
"status": “success”,
"data": []
}

5.2.8 Checkout

Guest can request checkout through TV which is in the room. PCS sends the balance to PMS to verify that the guest has confirmed the correct billing information. Refer the Figure 7.

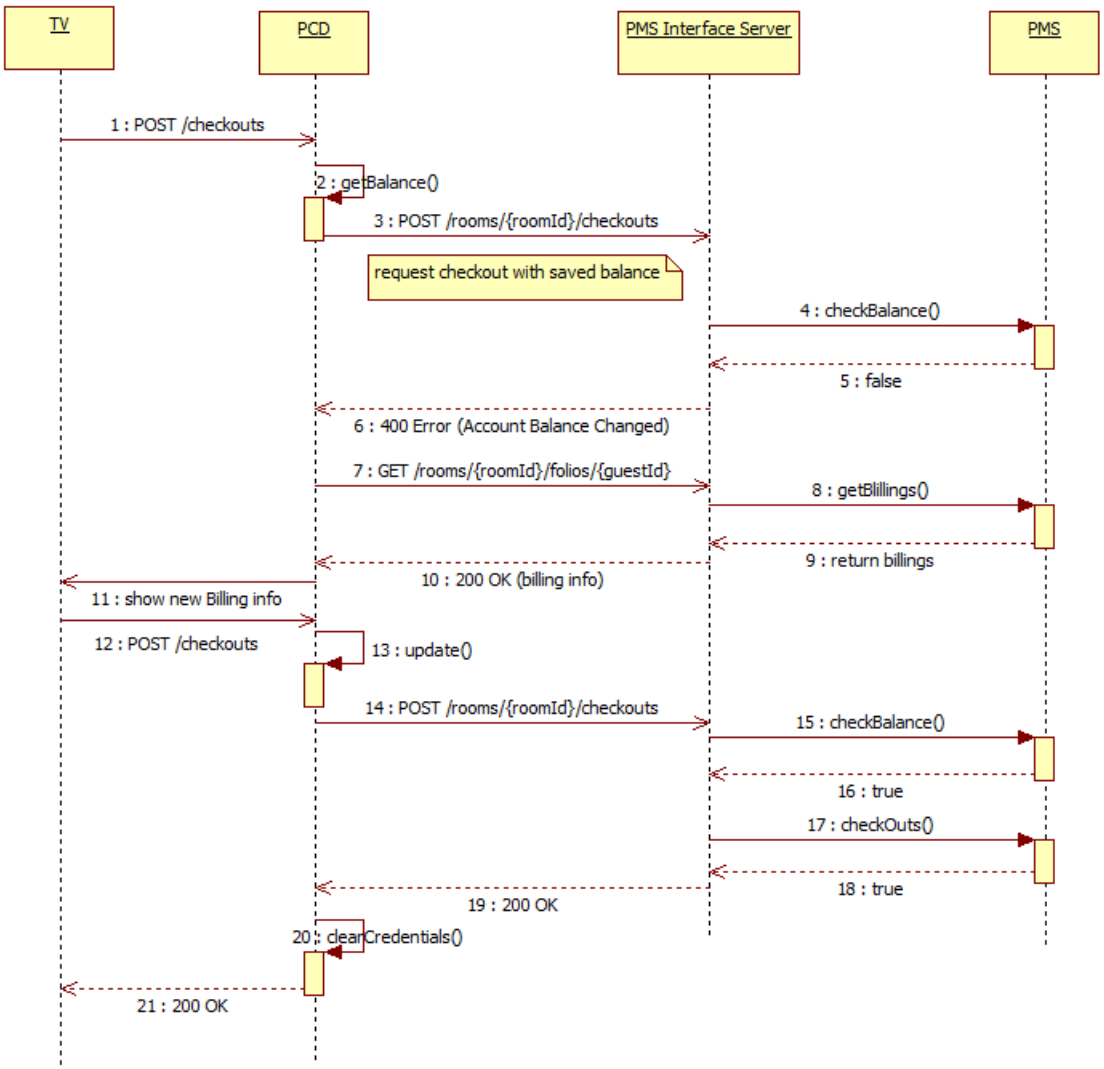

Figure 7

5.2.8.1 *POST /rooms/{room_id}/checkouts*

POST /rooms/{room_id}/checkouts API sends room checkout request to PMS server. Room checkout will checkout all guests in the same room at once.

5.2.8.1.1 Request: POST /rooms/{room_id}/checkouts

1. Headers
   - 1.1 auth
     - 1.1.1 client_id
     - 1.1.2 client_secret

5.2.8.1.2 Example

Headers
{

```
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        }
}
```

5.2.8.1.3 Responses

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Content-Type: application/json
- 1.2 Body
  - 1.2.1 status

2 401 (Unauthorized)

3 404 (Not Found) Folio Not Found

5.2.8.1.4 Example

```
{
        “statis” : “success”
}
```

5.2.8.2 POST /rooms/{room_id}/checkouts/{guest_id}

POST /rooms/{room_id}/checkouts/{guest_id} API sends a guest checkout request to PMS server. Request body must contain balance value which should match the balance of the guest folio.

5.2.8.2.1 Requests

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret

2 Body
- 2.1 balance

5.2.8.2.2 Example

Headers

```
{
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        }
}
```

Body

```
{
        “balance” : 18.8
}
```

5.2.8.2.3 Responses

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Content-Type: application/json
- 1.2 Body

1.2.1 status
2 401 (Unauthorized)
3 404 (Not Found)

5.2.8.2.4 Example

```
{
    “status” : “success”
}
```

5.2.9 Subscriptions

Subscribe to the PMS interface server for receiving the events. It does not require the poll if this subscription function is supported. Input 0 seconds in the polling time at External server setting page of Admin Client.

Attribute	Type	Description
name	String	Subscriber name
callbackUri	String	Subscribe to the External Server to receive events

5.2.9.1 POST /subscriptions

Register PCS’s API to receive PMS events. This api should be provided by External server side.

5.2.9.1.1 Requests : POST /api/pms/v2/subscriptions

1 Headers
1.1 Content-Type: application/json
1.2 auth
1.2.1 client_id
1.2.2 client_secret
2 Body
2.1.1 name
2.1.2 callbackUri

5.2.9.1.2 Example

Headers

```
{
    "auth": {
        " client_id": "123456789",
        " client_secret": "lge"
    }
}
```

Body

```
{
    “name”: “PMS”,
    "callbackUri ": " https://procentric.lge.com:60080/api/v2/events/pms
}
```

5.2.9.1.3 Responses

1 200 (OK)
1.1 Headers

1.1.1 Content-Type: application/json
1.2 Body
1.2.1 status
1.2.2 data
1.2.2.1 id
1.2.2.2 created
1.2.2.3 name
1.2.2.4 callbackUri
2 401 (Unauthorized)
3 404 (Not Found)

5.2.9.1.4 Example


```
{
    “status” : “success”,
    "data": {
            "id": 1,
            "created": "2017-04-03T07:49:07+00:00",
            "name": "PCN",
            "callbackUri": "https://procentric.lge.com:60080/api/v2/events/pms"
    }
}
```


5.2.9.2 GET /subscriptions/{subscription_id}

Retrieve a subscription information

5.2.9.2.1 Request : GET /api/pms/v2/subscriptions/{subscription_id}

1. Headers
1.1 auth
1.1.1 client_id
1.1.2 client_secret

5.2.9.2.2 Example

Headers


```
{
    "auth": {
            " client_id": "123456789",
            " client_secret": "lge"
    }
}
```


5.2.9.2.3 Responses

1 200 (OK)
1.1 Headers
1.1.1 Content-Type: application/json
1.2 Body
1.2.1 status
1.2.2 data
1.2.2.1 id
1.2.2.2 name
1.2.2.3 created
1.2.2.4 callbackUri

2 401 (Unauthorized)
3 404 (Not Found)

5.2.9.2.4 Example


```
{
	“status”: “success”,
	“data”: {
		"id":27,
		“name”: “PCN”,
		"created": "2017-05-26T00:30:08.138Z",
		"callbackUri": https://localhost:60080/api/v2/events/pms
	}
}
```


5.2.9.3 *PUT /subscriptions/{subscription_id}*

Update the subscription for events

5.2.9.3.1 Request : PUT /api/pms/v2/subscriptions/{subscription_id}

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret

2 Body
- 2.1 name
- 2.2 callbackUri
- 2.3 callbakToken

5.2.9.3.2 Example

Headers


```
{
	"auth": {
		" client_id": "123456789",
		" client_secret": "lge"
	}
}
```


Body


```
{
	“name”: “PMS”,
	"callbackUri ": " https://procentric.lge.com:60080/api/v2/events/pms"",
	“callbakToken”: “eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.ImRZM0FzeG9XVTh5Qkh6M3hWTHpsQXk”
}
```


5.2.9.3.3 Responses

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Content-Type: application/json
- 1.2 Body
  - 1.2.1 status
  - 1.2.2 data
    - 1.2.2.1 id
    - 1.2.2.2 name

1.2.2.3 created
1.2.2.4 updated
1.2.2.5 callbackUri

2 401 (Unauthorized)
3 404 (Not Found)

5.2.9.3.4 Example


```
{
    “status”: “success”,
    “data”: {
            "id":27,
            "created": "2017-05-26T00:30:08.138Z",
            "updated": "2017-05-27T00:30:08.138Z",
            "callbackUri": https://localhost:60080/api/v2/events/pms
    }
}
```


5.2.9.4 DELETE /subscriptions/{subscription_id}

Delete the subscription which is registered in PMS interface server

5.2.9.4.1 Request : DELETE /api/pms/v2/subscriptions/{subscription_id}

1. Headers
   1.1 auth
      1.1.1 client_id
      1.1.2 client_secret

5.2.9.4.2 Example

Headers


```
{
    "auth": {
            " client_id": "123456789",
            " client_secret": "lge"
    }
}
```


5.2.9.4.3 Responses

1 200 (OK)
   1.1 Headers
      1.1.1 Content-Type: application/json
   1.2 Body
      1.2.1 status
2 401 (Unauthorized)
3 404 (Not Found)

5.2.9.4.4 Example


```
{
    “status” : “success”,
    “data”: 1
}
```

5.2.10 Events Nofitication to PCS

Events refers to a time-ordered collection of events. The events has collections of all event type.
Refer the Figure 8

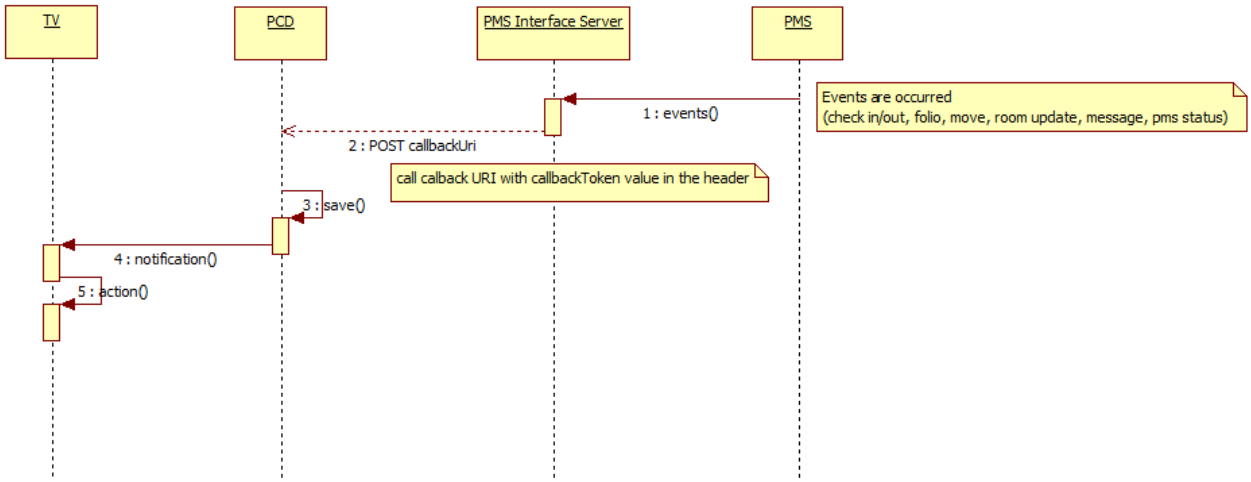

Figure 8

HEADER

Attribute	Type	Description
token	String	Token value issued from PCS via POST/PUT subscription, see the Authentication section.

BODY

Attribute	Type	Description
id	Integer	Event identifier
created	String	Created date of the event
Type	String	Event type. This attribute must be included in all events. ● checkin ● checkout ● move : change the room ● update : room information is updated ● popup : messages for the guest ● status : pms status
event objects	object	Room number

5.2.10.1 POST /events/pms

Retrieve all events. This api is provide by PCS to call event notification. Subscribe to the PMS interface server for receiving the events. This api would be callbackUrl's value as the parameter of subscriptions.

- **Check out event** process is different with other event because of multi guests possibility. When PCD receives check out messages from PMS, PCD calls GET /rooms/{room_id} api to check another guest is existed in the room. If PMS returns empty array for the room, PCD clears the data in the PCD finally.
- **Status event** is to notify PCD the PMS interface server is working properly or not.

5.2.10.1.1 Requests

1 Headers
- 1.1 Content-Type: application/json
- 1.2 token

2 Body
- 2.1 Data
  - 2.1.1 events – array
    - 2.1.1.1 id
    - 2.1.1.2 created
    - 2.1.1.3 type
    - 2.1.1.4 each event object

5.2.10.1.2 Example

```
curl -X POST https://pcs.lge.com:60080/api/v2/events/pms
-H 'Authorization: Bearer WVFJMzlMcmVBWjJmSU1EdHJjeU13bU9OVFZHQmNFc2FqdemDhjS8bM8Ghq4EgdAxPtafPQ' \
-H 'Content-Type: application/json' \
-d '{
        "data": {
                "events": [{
                        "id": 502,
                        "created": "2017-05-07T20:10:39.380Z",
                        "type": "checkin",
                        "checkin": {
                                "id": 502,
                                "room": "201",
                                "guest": "1234",
                                "payment": "cash",
                                "name": {
                                        "first": "Bilbo",
                                        "last": "o Baggins",
                                        "title": null
                                },
                                "e-mail": "bbaggins@gmail.com",
                                "source": {
                                        "type": null
                                },
                                "group": {
                                        "number": "12382734",
                                        "code": "Baggins Board Meeting"
                                },
                                "language": "en",
                                "vip_status": "1",
                                "no_post": true,
                                "affinity": {
                                        "member": "6547983218919",
                                        "status": "GOLD"
                                }
                        }
                },
                {
                        "id": 15426,
                        "created": "2015-10-02T14:36:53.884Z",
                        "type": "checkout",
```

```
                "checkout": {
                        "id": 12354,
                        "room": "101",
                        "guest": "101_1"
                }
        },
        {
                "id": 15424,
                "created": "2015-10-02T14:36:53.884Z",
                "type": "move",
                "move": {
                        "id": 12354,
                        "room": {
                                "from": "101",
                                "to": "102"
                        },
                        "guest": "101-1"
                }
        },
        {
                "id": 15423,
                "created": "2015-10-02T14:36:53.884Z",
                "type": "update",
                "update": {
                        "id": 12354,
                        "room": "101",
                        "guest": "101-1",
                        "payment": "credit",
                        "name": {
                                "title": null,
                                "last": null,
                                "first": null,
                                "middle": null,
                                "prefix": null,
                                "suffix": null
                        },
                        "email": "guest001@lge.com",
                        "group": {
                                "number": "g0001",
                                "code": "conference"
                        },
                        "language": "eng",
                        "vipStatus": "vip",
                        "affinity": {
                                "member": "LGE1234",
                                "program": "LGE-pr-0001",
                                "status": "gold"
                        }
                }
        },
        {
                "id": 15422,
                "created": "2015-10-02T14:36:53.884Z",
                "type": "popup",
```

```
                    "popup": {
                        "id": 12354,
                        "room": "101",
                        "type": "popup",
                        "message": "Hello"
                    }
                },
                {
                  "id": 15426,
                  "created": "2015-10-02T14:36:53.884Z",
                  "type": "status",
                  "status": {
                      "status": "up",              // up | down
                      "id": 15426                  // event ID
                   }
                }
            ]
        }
}'
```

5.3 API Client – QMS

QMS(Quality Management System) is provided to improve customer's satisfaction to the hotel. A guest can request some services to the hotel thru this system. In case of PCD, a guest uses TV.
PCD classifies several types according to service as follows.

Type	Code	Description
Amenity		It is used when a guest requests necessary items to the hotel such as towel.
	Defined by QMS	Each item is defined by QMS and varies from hotel to hotel. The list of items provided is received from QMS at initialization. Refer the 5.3.3 section
Wakeupcall		If a guest sets wakeup time, TV or hotel's staff acts to wake the guest.
	TV	TV turns on at the wakeup time automatically.
	QMS	A staff in the hotel calls the room to wake up a guest.
Reservation		Hotel provides some various services or activities. When a guest want to take part in the activies, a guest can reserve it thru this.
	Shuttle	To reserve shuttle bus.
Service		A guest can request some hotel's services.
	Laundry	Hotel provides laundry service and a guest can use this.
	DND_MUR	A guest can request DND(Do Not Disturb) or MUR(Make Up Room).

5.3.1 Prefix

It is defined prefix to call QMS APIs.

/api/qms/v3

5.3.2 Check

Check the connection between PCS and QMS Interface Server. verify connection with QMS server

5.3.2.1 POST /check

When an admin sets the QMS in the admin page, PCD checks the connection with QMS.

5.3.2.1.1 Requests : POST /api/qms/v3/check

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret

2 Body
- 2.1 name
- 2.2 host
- 2.3 port
- 2.4 isUse
- 2.5 isSSL : http / https
- 2.6 auth : object to authorize

5.3.2.1.2 Example

Headers
```
{
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        }
}
```
Body
```
{
        "name": "QMS",
        "host": "123.1.1.123",
        "port": "8881",
        "isUse": 1,
        "isSSL": 0,
        "auth": {
                "client_id": "0123456",
                "client_secret": "ABCD1234"
        }
}
```

5.3.2.1.3 Responses

1 200 (OK) : The verification is success
2 401 (Unauthorized) : fail to authorize, it should be checked client_id and client_secret values.
3 404 (PMS has no data) : check the host IP and port.

5.3.3 Services

The representation of amenity services of the site.
Each item is defined by QMS and varies from hotel to hotel. The list of items provided is received from QMS at initialization by calling services API.

Attribute	Type	Description

code	string	Concierge service code.
name	string	Concierge service name.
type	string	Service types (amenity | channel | laundry | shuttle |
description	string	Description for the service, optional value

5.3.3.1 GET /services

Retrieve concierge service list.

5.3.3.1.1 Requests : GET /api/qms/v3/services

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret

5.3.3.1.2 Example

Headers
```
{
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        }
}
```

5.3.3.1.3 Responses

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Cache-Control: no-cache
- 1.2 Body
  - 1.2.1 code : mandatory
  - 1.2.2 name : mandatory
  - 1.2.3 type : mandatory
  - 1.2.4 description : optional

2 401 (Unauthorized)

5.3.3.1.4 Example

```
[
    {
        "code":"449",
        "name":"Comb",
        "type":"amenity"
    },
    {
        "code":"451",
        "name":"Dental Kit",
        "type":"amenity"
    },
    {
      "code": "552",
      "name": "Shampoo",
```

```
    "type": "amenity"
    "description": {"price":"$0.00", "display":"It's free"}
  },
  {
      "code":"A_channel",
      "name":"sports channel",
      "type":"channel",
      "description": {"display":"sports package (ch11, ch12, ch13)", "price","$11.99"}
  },
  {
      "code":"B",
      "name":"movie channel",
      "type":"channel",
      "description": {"display":"movie package (ch21, ch22, ch23)", "price","$21.99"}
  },
  {
      "code":"B",
      "name":"laundry (Basic)",
      "type":"laundry",
      "description": {"price":{"pants":"$0.99", "shirts":"$1.99"}, "display":"within 24 hours"},
  {
      "code":"E",
      "name":"laundry (Express)",
      "type":"laundry",
      "description": {"price":{"pants":"$1.99", "shirts":"$2.99"}, "display":"within 12 hours"}
  },
  {
      "code":"1",
      "name":"Shuttle bus (Route 1)",
      "type":"shutttle",
      "description": {"price": {"adult":"$2.99", "child":"$1.00"}, "display":{"1":"08:00", "2":"09:00",….}}
  },
  {
      "code":"2",
      "name":"Shuttle bus (Route 2)",
      "type":"shutttle",
      "description": {"price": {"adult":"$1.99", "child":"$1.00"}, "display":{"1":"08:00", "2":"09:00",….}}
  }
]
```


5.3.4 Directories

The representation of service directory informations. It can be noticed hotel's facility information such as restaurant. There is no uses in PCD currently, so return empty string is OK.

Attribute	Type	Description
id	Integer	Directory identifier.
site_id	string	Hotel code, use client_id value
Type	string	Hotel email address.
description	string	contains HTML data describing the directory
title	string	Directory title
image	string	Image URI

5.3.4.1 GET /directories

Retrieve service directories

5.3.4.1.1 Resquest : GET /api/qms/v3/directories

1 Headers
- 1.1 Cache-Control: no-cache
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret

5.3.4.1.2 Example

Headers
```
{
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        }
}
```

5.3.4.1.3 Responses

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Cache-Control: no-cache
- 1.2 Body
  - 1.2.1 id
  - 1.2.2 site_id
  - 1.2.3 type
  - 1.2.4 description
  - 1.2.5 title
  - 1.2.6 image

2 401 (Unauthorized)

5.3.4.1.4 Example

```
[
        {
                "id":20,
                "site_id":"123456789",
                "description":"<strong>Luggage Assistance</strong><br/>For any luggage assistance, please contact the Guest Services Centre.<br/><strong>Taxi Service</strong><br/>Should you require a taxi, please contact the Guest Services Centre to arrange one for you.<br/><strong>Courier Service</strong><br/>Worldwide courier service is available; please contact the Guest Services Centre to assist you.<br/><strong>Airport Transfer and Limousine Services</strong><br/>Arrangements may be made for the airport transfers and city trips. Please contact the Guest Services Centre&amp;nbsp;for information and reservations.<br/>",
                "image":"img/service_directory/images/S1397732218760.jpg?1397732218760",
                "title":"Concierge Services"
        }
]
```

5.3.5 Requests

The representation of requests of concierge services in a room.

5.3.5.1 Request Entity

5.3.5.1.1 Request Body

Attribute	Type	Description
request_type	string	Requests' categories (amenity, wakeupcall, reservation, service, channel)
request_code	string	An unique service item code
request_name	string	Display name of amenity service (ex. towel)
request_description	object	Json string, description for the request such as count.

5.3.5.1.2 Response

Attribute	Type	Description
request_id	string	An unique request id
room_id	string	Requested room id
account_id	string	A guest id
status	string	Request's status: ● N: New Work Request ● A: Assigned ● F: Job Done ● X: Cancelled ● D: Deferred
request_type	string	Requests' categories (amenity, wakeupcall, reservation, service, channel)
request_code	string	An unique service item code
request_name	string	Display name of amenity service (ex. towel)
request_description	object	Json string, descripition for the request such as count.

5.3.5.2 POST /requests/{room_id}

Create new requests.

5.3.5.2.1 Request

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret

2 Body
- 2.1 request_type : (String) Request' categories (amenity, wakeupcall, service, channel)
- 2.2 request_code : (String) An unique service item code
- 2.3 request_name : (String) display name of concierge services (ex. towel)
- 2.4 request_description : (Object) descriptions for the request such as count

5.3.5.2.2 Response

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Cache-Control: no-cache
- 1.2 Body

1.2.1 request_id : (String) An unique request id
1.2.2 room_id : (String) requested room id
1.2.3 account_id : (String) a guest id
1.2.4 status : (String) request's status (N : New, A : Assigned, F : Job Done, X : Cancelled, D : Deferred)
1.2.5 request_type : (String) Request' categories (amenity, wakeupcall, service, channel)
1.2.6 request_code : (String) An unique service item code
1.2.7 request_name : (String) display name of concierge services (ex. towel)
1.2.8 request_description : (Object) descriptions for the request such as count

5.3.5.2.3 Example

1 Amenity

A guest request to get amenity services. The amenity list is getting from /services API.

1.1 Request

resource	POST http://qms.lge.com:9888/api/qms/v3/requests/201
header	Content-Type: application/json auth: {"client_id":"0123456", "client_secret":"ABCD1234"}

body	{
	"account_id": "201_a",
	"request_type": "amenity",
	"request_code": "430",
	"request_name": "towel",
	"request_description": {
	"count": 3
	}
	}

1.2 Response

status	200 OK

body	{
	"request_id": " DLqD5AOhG1Q_eXSd ",
	"room_id": "201",
	"account_id": "201_a",
	"status": "N",
	"request_type": "amenity",
	"request_code": "430",
	"request_name": "towel",
	"request_description": {
	"count": 3
	}
	}

2 Wakeupcall

There are 2 kinds of wakeupcall function by calling or TV. It can be distinguished by "request_code".

2.1 Request

resource	POST http://qms.lge.com:9888/api/qms/v3/requests/201
header	Content-Type: application/json auth: {"client_id":"0123456", "client_secret":"ABCD1234"}

Body (QMS)	{
	"account_id": "201_a",
	"request_type": "wakeupcall",
	"request_code": "QMS",
	"request_name": "wakeup call",
	"request_description": {
	"time": "07:30"
	}
	}

Body (TV)	{
	"account_id": "201_a",
	"request_type": "wakeupcall",
	"request_code": "TV",
	"request_name": "wakeup call",
	"request_description": {
	"time": "07:30",
	"waiting_munute": 30,
	"isNotification": true
	}
	}

2.2 Response

status	200 OK

body	{
	"request_id": "DLqD5AOhG1Q_eXSd",
	"room_id": "201",
	"account_id": "201_a",
	"status": "N",
	"request_type": "wakeupcall",
	"request_code": "QMS",
	"request_name": "wakeup call",
	"request_description": {
	"time": "07:30"
	}
	}

3 Reservation

Reservation requests like shuttle bus, restaurant.

3.1 Request

resource	POST http://qms.lge.com:9888/api/qms/v3/requests/201
header	Content-Type: application/json auth: {"client_id":"0123456", "client_secret":"ABCD1234"}

body	{
	"account_id": "201_a",
	"request_type": "reservation",
	"request_code": "shuttle",
	"request_name": "shuttle bus",
	"request_description": {
	"code": "1",
	"date": "yyyy-mm-dd",
	"time": "07:30",
	"adult": 2,
	"child": 1
	}
	}

3.2 Response

status	200 OK
body	{ "request_id": "DLqD5AOhG1Q_eXSd", "room_id": "201",

	"account_id": "201_a",
	"status": "N",
	"request_type": "reservation",
	"request_code": "shuttle",
	"request_name": "shuttle bus",
	"request_description": {
	"code": "1",
	"time": "07:30",
	"adult": 2,
	"child": 1
	}
	}

4 Service

This requests are included for service requests such as laundry, make up room.

4.1 Request

resource	POST http://qms.lge.com:9888/api/qms/v3/requests/201
header	Content-Type: application/json auth: {"client_id":"0123456", "client_secret":"ABCD1234"}

body	{
	"account_id": "201_a",
	"request_type": "service",
	"request_code": "laundry",
	"request_name": "laundry", // optional
	"request_description": {
	"code": "E"
	}
	}

body (DND/MUR)	{
	"account_id": "201_a",
	"request_type": "service",
	"request_code": "DND_MUR",
	"request_name": "DND MUR",
	"request_description": {
	"code": "DND" // code : DND, MUR, NONE
	}
	}

4.2 Response

status	200 OK

body	{
	"request_id": "DLqD5AOhG1Q_eXSd",
	"room_id": "201",
	"account_id": "201_a",
	"status": "N",
	"request_type": "service",
	"request_code": "laundry",
	"request_name": "laundry",
	"request_description": {
	"code": "E"
	}
	}

body (DND/MUR)	{
	"request_id": "DLqD5AOhG1Q_eXSd", // valid only first assigned id from QMS
	"room_id": "201",
	"account_id": "201_a",
	"status": "N",
	"request_type": "service",
	"request_code": "DND_MUR",
	"request_name": "DND MUR",
	"request_description": {
	"code": "DND"
	}
	}

5 Channel

A guest can buy paid channel or channel packages.

5.1 Request

resource	POST http://qms.lge.com:9888/api/qms/v3/requests/201
header	Content-Type: application/json auth: {"client_id":"0123456", "client_secret":"ABCD1234"}

body	{
	"account_id": "201_a",
	"request_type": "channel",
	"request_code": "A",
	"request_name": "sports channel (ch 11, ch12, ch13)",
	"request_description": {
	"code": "A"
	}
	}

5.2 Response

status	200 OK

body	{
	"request_id": "DLqD5AOhG1Q_eXSd",
	"room_id": "201",
	"account_id": "201_a",
	"status": "N",
	"request_type": "channel",
	"request_code": "A",
	"request_name": "sports channel (ch 11, ch12, ch13)",
	"request_description": {
	"code": "A"
	}
	}

5.3.5.3 PUT /requests/{room_id}

PCD calls requests API with PUT method to modify the previous request such as cancelling requests or changing request's descriptions.

5.3.5.3.1 Request

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret

2 Body
- 2.1 request_id : (String) the unique request id which is received from QMS at first request.
- 2.2 status : (String) When a guest want to cancel the previous request, the status value is "X".
- 2.3 request_description : (Object) descriptions for the request such as count

The status and request_description cannot be changed simutaniously.

5.3.5.3.2 Response

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Cache-Control: no-cache
- 1.2 Body
  - 1.2.1 request_id : (String) An unique request id
  - 1.2.2 room_id : (String) requested room id
  - 1.2.3 account_id : (String) a guest id
  - 1.2.4 status : (String) request's status (N : New, A : Assigned, F : Job Done, X : Cancelled, D : Deferred)
  - 1.2.5 request_type : (String) Request' categories (amenity, wakeupcall, service, channel)
  - 1.2.6 request_code : (String) An unique service item code
  - 1.2.7 request_name : (String) display name of concierge services (ex. towel)
  - 1.2.8 request_description : (Object) descriptions for the request such as count

5.3.6 Receiving check in / out events

The QMS server can get check in / out events from PCD if the notification events option is checked in the external server configuration page at Admin Client. When PCD receives the events from PMS, PCD notifies the events to the QMS server. The QMS server must provide below APIs to get the events from PCD.

5.3.6.1 Delivery check in events

Notify the room's status to QMS when a guest check in. PCD calls "POST /rooms/{room_id}" api for check in notification.

5.3.6.1.1 Requests : POST /api/qms/v3/rooms/{room_id}

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret

2 Body
- 2.1 id
- 2.2 groupCode
- 2.3 deploymentId
- 2.4 tvs (array)
- 2.5 guests (array)

5.3.6.1.2 Example

Headers
```
{
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        }
}
```
Body
```
{
  "id": "201",          // room ID
  "groupCode": "0",
  "deploymentId": 1,
```

```
  "tvs": [{
    "tvRoomNumber": "201_a",   // tv ID
    "xait": "ver1",
    "serial": "123SERIAL00A",
    "model": "POST-5500-MAN",
    "platformVersion": "01.02.30.04",
    "micomVersion": "m1.2.3.4",
    "ip": "127.0.0.1",
    "mac": "AB:CD:EF:GH",
    "powerMode": "on",
    "network": "connect",
    "insertAt": "2018-01-25 04:19:03",
    "updateAt": "2018-03-18 14:09:51",
    "roomId": "201",
    "epgFetchDate": "2017-01-test",
    "projectVersion": "1.0",
    "updateStat": "complete"
  }],
  "guests": [{
    "id": "201-1",        // guest ID
    "language": "en",
    "email": "bbaggins@gmail.com",
    "phone": "",
    "vipStatus": "1",
    "payment": "cash",
    "checkin": "2017-05-07 20:10:39",
    "title": null,
    "fullName": "Bilbo  o'Baggins",
    "lastName": "o'Baggins",
    "firstName": "Bilbo",
    "middleName": "",
    "prefix": "",
    "suffix": "",
    "affinityProgram": "",
    "affinityMember": "6547983218919",
    "affinityStatus": "GOLD",
    "groupNumber": "12382734",
    "groupCode": "Baggin's Board Meeting",
    "insertAt": "2018-03-27 23:49:30",
    "updateAt": "2018-03-27 23:49:30",
    "roomId": "201"
  }]
}
```


5.3.6.1.3 Responses

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Content-Type: application/json
- 1.2 Body
  - 1.2.1 status

2 401 (Unauthorized)

3 404 (Not Found)

5.3.6.1.4 Example

```
{
        "status": “success”,
}
```

5.3.6.2 Delivery check out events

When PCD receives the check out events, PCD notifies the event to QMS server by calling “DELETE /rooms/{room_id}” API4. After check out the room, QMS server should not send the pay channel list to PCD until the room check in again.

5.3.6.2.1 Request : DELETE /api/qms/v3/rooms/{room_id}

1. Headers
   - 1.1 auth
     - 1.1.1 client_id
     - 1.1.2 client_secret

5.3.6.2.2 Example

```
Headers
{
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        }
}
```

5.3.6.2.3 Responses

1 200 (OK)
   - 1.1 Headers
     - 1.1.1 Content-Type: application/json
   - 1.2 Body
     - 1.2.1 status

2 401 (Unauthorized)
3 404 (Not Found)

5.3.6.2.4 Example

```
{
        “status” : “success”,
}
```

5.3.7 Subscriptions

Subscribe to the QMS server for receiving the events. It does not require the poll if this subscription function is supported. QMS interface server call the registered callback URI when the status of guest’s request is changed.

Attribute	Type	Description
name	String	Subscriber name
callbackUri	String	Subscribe to the External Server to receive events

5.3.7.1 POST /subscriptions

Register PCS's API to receive QMS events

5.3.7.1.1 Requests : POST /api/qms/v3/subscriptions

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret

2 Body
- 2.1.1 name
- 2.1.2 callbackUri

5.3.7.1.2 Example

Headers
```
{
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        }
}
```
Body
```
{
        "name": "QMS",
        "callbackUri": "https://procentric.lge.com:60080/api/v3/events/qms"
}
```

5.3.7.1.3 Responses

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Content-Type: application/json
- 1.2 Body
  - 1.2.1 status
  - 1.2.2 data
    - 1.2.2.1 id
    - 1.2.2.2 created
    - 1.2.2.3 name
    - 1.2.2.4 callbackUri

2 401 (Unauthorized)

3 404 (Not Found)

5.3.7.1.4 Example

```
{
        "status" : "success",
        "data": {
                "id": 1,
                "created": "2017-04-03T07:49:07+00:00",
                "name": "QMS",
                "callbackUri": "https://procentric.lge.com:60080/api/v3/events/qms"
        }
}
```

5.3.7.2 *GET /subscriptions/{subscription_id}*

Retrieve a subscription information

5.3.7.2.1 Request : GET /api/qms/v3/subscriptions/{subscription_id}

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret

5.3.7.2.2 Example

Headers
```
{
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        }
}
```

5.3.7.2.3 Responses

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Content-Type: application/json
- 1.2 Body
  - 1.2.1 status
  - 1.2.2 data
    - 1.2.2.1 id
    - 1.2.2.2 name
    - 1.2.2.3 created
    - 1.2.2.4 callbackUri

2 401 (Unauthorized)
3 404 (Not Found)

5.3.7.2.4 Example

```
{
        “status”: “success”,
        “data”: {
                "id":27,
                “name”: “QMS”,
                "created": "2017-05-26T00:30:08.138Z",
                "callbackUri": https://procentric.lge.com:60080/api/v3/events/qms
        }
}
```

5.3.7.3 *PUT /subscriptions/{subscription_id}*

Update the subscription for events

5.3.7.3.1 Request : PUT /api/qms/v3/subscriptions/{subscription_id}

1 Headers
- 1.1 Content-Type: application/json

1.2 auth
1.2.1 client_id
1.2.2 client_secret
2 Body
2.1 name
2.2 callbackUri

5.3.7.3.2 Example

Headers
{
"auth": {
" client_id": "123456789",
" client_secret": "lge"
}
}
Body
{
“name”: “QMS”,
"callbackUri ": "https://procentric.lge.com:60080/api/v3/events/qms
}

5.3.7.3.3 Responses

1 200 (OK)
1.1 Headers
1.1.1 Content-Type: application/json
1.2 Body
1.2.1 status
1.2.2 data
1.2.2.1 id
1.2.2.2 name
1.2.2.3 created
1.2.2.4 updated
1.2.2.5 callbackUri
2 401 (Unauthorized)
3 404 (Not Found)

5.3.7.3.4 Example

{
“status”: “success”,
“data”: {
"id":27,
"created": "2017-05-26T00:30:08.138Z",
"updated": "2017-05-27T00:30:08.138Z",
"callbackUri": https://localhost:60080/api/v2/events/qms
}
}

5.3.7.4 DELETE /subscriptions/{subscription_id}

Delete the subscription which is registered in QMS interface server

5.3.7.4.1 Request : DELETE /api/qms/v3/subscriptions/{subscription_id}

1 Headers
  1.1 Content-Type: application/json
  1.2 auth
    1.2.1 client_id
    1.2.2 client_secret

5.3.7.4.2 Example

Headers
{
  "auth": {
    " client_id": "123456789",
    " client_secret": "lge"
  }
}

5.3.7.4.3 Responses

1 200 (OK)
  1.1 Headers
    1.1.1 Content-Type: application/json
  1.2 Body
    1.2.1 status
2 401 (Unauthorized)
3 404 (Not Found)

5.3.7.4.4 Example

{
  “status” : “success”,
  “data”: 1
}

5.3.8 Events Nofitication to PCS

Events refers to a time-ordered collection of events. The events has collections of all event type.

HEADER

Attribute	Type	Description
token	String	Token value issued from PCS via POST/PUT subscription

BODY

Attribute	Type	Description
id	Integer	Event identifier
created	String	Created date of the event
Type	String	Event type. This attribute must be included in all events. ● directory : if it was changed any, all list should be sent. ● service : if it was changed any, all list should be sent. ● request
event objects	object	Room number

5.3.8.1 POST /events/qms

Retrieve all events. This api is provide by PCS to call event notification such as changing status of the guest's request or services. Subscribe to the QMS interface server for receiving the events. This api would be callbackUrl's value as the parameter of subscriptions.

5.3.8.1.1 Requests

1 Headers
- 1.1 Content-Type: application/json
- 1.2 token

2 Body
- 2.1 events – array
  - 2.1.1 id
  - 2.1.2 roomId
  - 2.1.3 created
  - 2.1.4 each event object
    - 2.1.4.1 type

5.3.8.1.2 Example

Resource	POST /api/v3/events/qms
Head	Authorization: 'Bearer {token}'
Params	{ "id": 1234, // event id (incremental, unique) "created": "2015-10-02T14:36:53.884Z", // event time "events": [{ "type": "request", // event type : request, service, directory "request_id": "DLqD5AOhG1Q_eXSd", "room_id": "201", "status": "A", // Assign, Finish, Defer "popup": "Your request is assigned." // optional, send to message module. }, { "type": "request", // event type : request, service, directory "request_id": "2LqD5AOhG1Q_eXSd", "room_id": "201", "status": "F" }, { "type": " request", "request_id": "2LqD5AOhG1Q_eXSd", "room_id": "201", "request_code": "DND_MUR", "status": "A", "description": { "DND": "ON", "MUR": "OFF" } }, { // based request information is changed (sending all service list) "type": "service", "created": "2015-10-02T14:36:53.884Z", "services": [{

```
                                        "code": "1",
                                        "name": "Shuttle bus (Route 1)",
                                        "type": "shutttle",
                                        "description": {
                                                "price": {
                                                        "adult": "$3.99",
                                                        "child": "$2.00"
                                                },
                                                "schedule": {
                                                        "1": "08:00",
                                                        "2": "09:00"
                                                }
                                        }
                                }]
                        },
                        {    // directories information change events
                                "id": 20,
                                "type": "directory",
                                "description": {
                                        "title": "LG restaurant",
                                        "open": "09:00",
                                        "close": "22:00"
                                },
                                "image": "TBD",
                                "title": "Restaurant"
                        }
                ]
}
```


5.4 API Client – Pay Channel

5.4.1 Pay Channel Server Configuration

An admin should input the pay channel server information at PCS's Admin Web Client. The menu is Setting → External Server. The input areas are changed as choosing the server name. A detailed description of the server settings can be found in the Check API(5.4.4 check) for each server.

Explanation about the input area:

- Use Button : Select ON or OFF for the server connection. If selection is On, PCS checks the connection to the external server.
- Type : Using SSL or not. It depends on external server security.
- Host :The External Interface Server IP address.
- Port : The External Interface Server port number.
- Noti. Event : Required if Pay Channel Server has PMS capabilities, the pay channel server does not need to be notified when a guest is checking in / out.
- Prefix : It can be changed default prefix(5.4.3 Prefix) when PCD calls Pay Channel server's API.
- Authorization Section : This section is for authentication or to distinguish hotels. See the 4.2 authentication section.

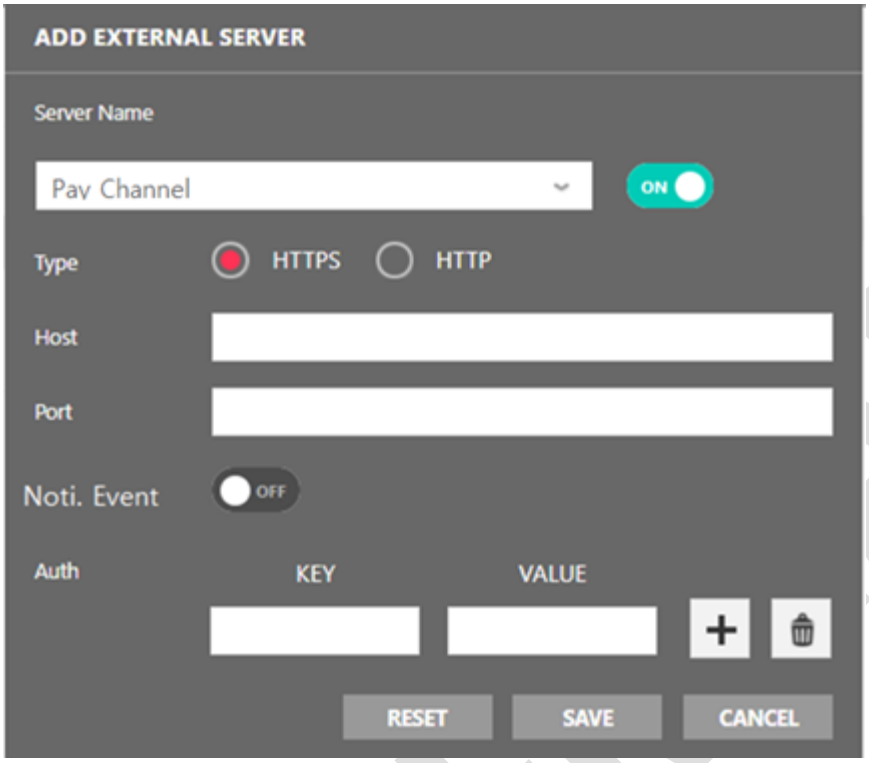

Figure 9

5.4.2 Scenarios to communicate with pay channel server

The basic sceinarios are below.

- Pay channel request event can be delivered only between Check in and out.
- TV will have a full channel list & default channel list in advance.
- Default channels can be changed only admin page without API. Only addition channel can be enabled by Pay channel request event.
- SI system will send a Pay channel request event to PCD and it will contain room number, paid channel list, expired date.
- Subscription base event communication.
- Paid channel should be updated without reboot.
- When payment duration expires, it will show default channel list back.
- Remaining time should be notified in advance.
- Admin page will show the pay channel information for each room with duration in detail information.
- When server is disconnected, paid channel should be expired correctly.

Refer the sequence diagram for basic sceinarios are below.

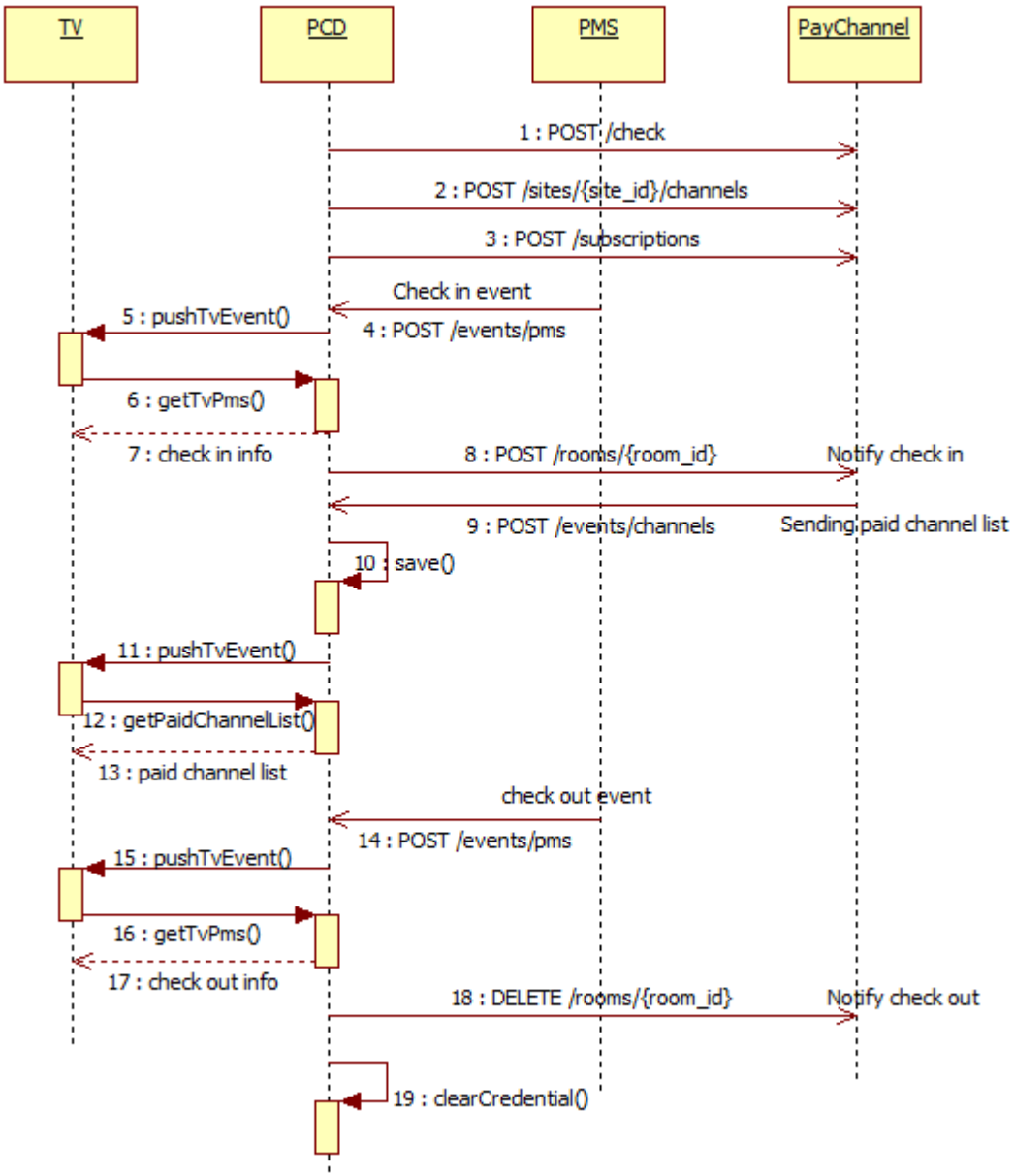

Figure 10

5.4.3 Prefix

It is defined prefix to call Pay channel APIs.

/api/v1

5.4.4 Check

Check the connection between PCS and Pay channel Interface Server. verify connection with Pay channel server

5.4.4.1 POST /check

Verify connection with the pay channel server.

5.4.4.1.1 Requests : POST /api/v1/check

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret
  - 1.2.3 … (All values in the auth section at PCD's External server setting page)

2 Body
- 2.1 user
- 2.2 password
- 2.3 auth

5.4.4.1.2 Example

Headers
```
{
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        }
}
```
Body
```
{
        "name": "PayChannel",
        "host": "1.2.3.45",
        "port": 8888,
        "isSSL": 0,
        "isNotifyEvent":true,
        "auth": {
        "client_id": "123456789",
        "client_secret": "lge"
        }
}
```

5.4.5 Channels

To manage channels and synchronous with PCD and pay channel server.

5.4.5.1 POST /channels

PCD requests all channel list from pay channel server.

5.4.5.1.1 Requests : POST /api/v1/sites/{site_id}/channels

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret

2 Body
2.1 Channels (array)
2.1.1 logicalChannelNumber : The unique channel information ID
2.1.2 type (optional) : IP or RF
2.2 other channel information (optional)

5.4.5.1.2 Example


```
{
  "channels":
  [
    {
      "logicalChannelNumber": 2,
      "type": "RF_MM",
      "streamType": 48,
      "encrypted": "true",
      "major": 13,
      "minor": 1,
      "logoFileName": "CBS_Logo.png",
      "channelMapListing": 5947,
      "channelCategory": "None",
      "channelLabel": "13-1 clear"                    // unique channel ID, primary key
    },
    {
      "logicalChannelNumber": 3,
      "type": "RF_MM",
      "streamType": 48,
      "encrypted": "true",
      "major": 26,
      "minor": 1,
      "logoFileName": "CNN_Logo.png",
      "channelMapListing": 429,
      "channelCategory": "None",
      "channelLabel": "26-1"
    },
    {
      "logicalChannelNumber": 4,
      "type": "IP",
      "streamType": 16,
      "encrypted": "true",
      "ipAddress": "227.0.0.0",
      "ipPort": 1234,
      "programNumber": 1,
      "PCR_PID": -1,
      "videoPID": -1,
      "videoType": -1,
      "audioPID": -1,
      "audioType": -1,
      "logoFileName": "AMC_Logo.png",
      "defaultEventLanguage": "en_US",
      "defaultEventTitle": "some%20default%20title%20goes%20here",
      "channelMapListing": 0,
      "channelCategory": "None",
      "channelLabel": "227.0.0.0:1234"
    }
```

```
  ]
}
```

5.4.5.2 GET /channels/{channel_id}

PCD requests the detailed channel information to pay channel server.

5.4.5.2.1 Requests : GET /api/v1/channels/4

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret

5.4.5.2.2 Example

```
Headers
{
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        }
}
```

5.4.5.2.3 Responses

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Cache-Control: no-cache
- 1.2 Body
  - 1.2.1 logicalChannelNumber : The unique channel information ID
  - 1.2.2 type (optional) : IP or RF
  - 1.2.3 other channel information (optional)

5.4.5.2.4 Example


```
{
   "logicalChannelNumber": 4,
   "type": "IP",
   "streamType": 16,
   "encrypted": "true",
   "ipAddress": "227.0.0.0",
   "ipPort": 1234,
   "programNumber": 1,
   "PCR_PID": -1,
   "videoPID": -1,
   "videoType": -1,
   "audioPID": -1,
   "audioType": -1,
   "logoFileName": "AMC_Logo.png",
   "defaultEventLanguage": "en_US",
   "defaultEventTitle": "some%20default%20title%20goes%20here",
   "channelMapListing": 0,
   "channelCategory": "None",
```

```
    "channelLabel": "227.0.0.0:1234"
}
```

5.4.6 Receiving check in / out events

The pay channel server can get check in / out events from PCD if the notification events option is checked in the external server configuration page at Admin Client. When PCD receives the events from PMS, PCD notifies the events to the pay channel server. The pay channel server provides below APIs to get the events from PCD.

5.4.6.1 POST /rooms/{room_id}

Notify new rooms information when a guest check in.

5.4.6.1.1 Requests : POST /api/v1/rooms/{room_id}

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret

2 Body
- 2.1 id
- 2.2 groupCode
- 2.3 deploymentId
- 2.4 tvs (array)
- 2.5 guests (array)

5.4.6.1.2 Example

Headers
```
{
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        }
}
```
Body
```
{
   "id": "201",         // room ID
   "groupCode": "0",
   "deploymentId": 1,
   "tvs": [{
      "tvRoomNumber": "201_a",   // tv ID
      "xait": "ver1",
      "serial": "123SERIAL00A",
      "model": "POST-5500-MAN",
      "platformVersion": "01.02.30.04",
      "micomVersion": "m1.2.3.4",
      "ip": "127.0.0.1",
      "mac": "AB:CD:EF:GH",
      "powerMode": "on",
      "network": "connect",
      "insertAt": "2018-01-25 04:19:03",
      "updateAt": "2018-03-18 14:09:51",
      "roomId": "201",
```

```
    "epgFetchDate": "2017-01-test",
    "projectVersion": "1.0",
    "updateStat": "complete"
  }],
  "guests": [{
    "id": "201-1",        // guest ID
    "language": "en",
    "email": "bbaggins@gmail.com",
    "phone": "",
    "vipStatus": "1",
    "payment": "cash",
    "checkin": "2017-05-07 20:10:39",
    "title": null,
    "fullName": "Bilbo  o'Baggins",
    "lastName": "o'Baggins",
    "firstName": "Bilbo",
    "middleName": "",
    "prefix": "",
    "suffix": "",
    "affinityProgram": "",
    "affinityMember": "6547983218919",
    "affinityStatus": "GOLD",
    "groupNumber": "12382734",
    "groupCode": "Baggin's Board Meeting",
    "insertAt": "2018-03-27 23:49:30",
    "updateAt": "2018-03-27 23:49:30",
    "roomId": "201"
  }]
}
```


5.4.6.1.3 Responses

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Content-Type: application/json
- 1.2 Body
  - 1.2.1 status

2 401 (Unauthorized)

3 404 (Not Found)

5.4.6.1.4 Example


```
{
    "status": “success”,
}
```


5.4.6.2 DELETE /rooms/{room_id}

When PCD receives the check out events, PCD notifies the event to pay channel server. After check out the room, Pay channel server should not send the pay channel list to PCD until the room check in.

5.4.6.2.1 Request : DELETE /api/v1/rooms/{room_id}

1. Headers
   1.1 auth
      1.1.1 client_id
      1.1.2 client_secret

5.4.6.2.2 Example

```
Headers
{
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        }
}
```

5.4.6.2.3 Responses

1 200 (OK)
   1.1 Headers
      1.1.1 Content-Type: application/json
   1.2 Body
      1.2.1 status
2 401 (Unauthorized)
3 404 (Not Found)

5.4.6.2.4 Example

```
{
        “status” : “success”,
}
```

5.4.7 Paid Channels

Synchronize the data on the PCD and paid channel server by checking for the channels that the guests in the room are currently purchasing.

5.4.7.1 GET /rooms/{room_id}/channels

PCD requests current paid channel list for the room to pay channel server.

5.4.7.1.1 Requests : GET /api/v1/rooms/{room_id}/channels

1 Headers
   1.1 Content-Type: application/json
   1.2 auth
      1.2.1 client_id
      1.2.2 client_secret

5.4.7.1.2 Example

```
Headers
{
        "auth": {
                " client_id": "123456789",
```

```
                " client_secret": "lge"
        }
}
```

5.4.7.1.3 Responses

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Cache-Control: no-cache
- 1.2 Body
  - 1.2.1 status
  - 1.2.2 data (object)
    - 1.2.2.1 roomId : room ID
    - 1.2.2.2 paidChannels (array)
      - 1.2.2.2.1 logicalChannelNumber
      - 1.2.2.2.2 expireDate
      - 1.2.2.2.3 (addition informations)

5.4.7.1.4 Example


```
{
  "status": "success",
  "data": {
    "roomId": "201",
    "checkin": true,
    "paidChannels": [
      {
        "logicalChannelNumber": 6,
        "type": "ATSC_3",
        "streamType": 81,
        "encrypted": "false",
        "major": 6,
        "minor": 1,
        "plpId": 1,
        "channelMapListing": "0",
        "channelLabel": "SBS",
        "channelCategory": "None",
        "siteId": "0219127",
        "_id": "NpaKe0M6LV3Zo9Ih",
        "expireDate": "2019-04-04 14:17:38"
      },
      {
        "logicalChannelNumber": 7,
        "type": "ATSC_3",
        "streamType": 81,
        "encrypted": "false",
        "major": 7,
        "minor": 1,
        "plpId": 1,
        "channelMapListing": "0",
        "channelLabel": "KBS2",
        "channelCategory": "None",
        "siteId": "0219127",
        "_id": "8il55cJcpmzjLWvH",
        "expireDate": "2019-04-04 14:19:19"
```

```
        }
      ],
      "_id": "pYJg4UTagSQsNjX7"
    }
}
```

5.4.8 Subscriptions

Subscribe to the pay channel server for receiving the events. It does not require the poll if this subscription function is supported. The pay channel interface server call the registered callback URI when the status of guest's purchased channel list is changed.

Attribute	Type	Description
name	String	Subscriber name
callbackUri	String	Subscribe to the External Server to receive events

5.4.8.1 POST /subscriptions

Registering PCS's API to receive channels events

5.4.8.1.1 Requests : POST /api/v1/subscriptions

1 Headers
  1.1 Content-Type: application/json
  1.2 auth
    1.2.1 client_id
    1.2.2 client_secret
2 Body
    2.1.1 name
    2.1.2 callbackUri

5.4.8.1.2 Example

```
Headers
{
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        }
}
Body
{
        "name": "Pay Channel",
        "callbackUri ": " https://procentric.lge.com:60080/api/v2/events/channels"
}
```

5.4.8.1.3 Responses

1 200 (OK)
  1.1 Headers
    1.1.1 Content-Type: application/json
  1.2 Body
    1.2.1 status
    1.2.2 data

1.2.2.1 id
1.2.2.2 created
1.2.2.3 name
1.2.2.4 callbackUri

2 401 (Unauthorized)
3 404 (Not Found)

5.4.8.1.4 Example


```
{
	“status” : “success”,
	"data": {
		"id": 1,
		"created": "2018-04-03T07:49:07+00:00",
		"name": "Pay Channel",
		"callbackUri": "https://procentric.lge.com:60080/api/v2/events/channels"
	}
}
```


5.4.8.2 GET /subscriptions/{subscription_id}

Retrieve a subscription information

5.4.8.2.1 Request : GET /api/v1/subscriptions/{subscription_id}

1 Headers
1.1 Content-Type: application/json
1.2 auth
1.2.1 client_id
1.2.2 client_secret

5.4.8.2.2 Example

Headers


```
{
	"auth": {
		" client_id": "123456789",
		" client_secret": "lge"
	}
}
```


5.4.8.2.3 Responses

1 200 (OK)
1.1 Headers
1.1.1 Content-Type: application/json
1.2 Body
1.2.1 Status
1.2.2 data
1.2.2.1 id
1.2.2.2 name
1.2.2.3 created
1.2.2.4 callbackUri
2 401 (Unauthorized)
3 404 (Not Found)

5.4.8.2.4 Example

```
{
	“status”: “success”,
	“data”: {
		"id":27,
		“name”: “Pay Channel”,
		"created": "2018-05-26T00:30:08.138Z",
		"callbackUri": "https://localhost:60080/api/v2/events/channels"
	}
}
```

5.4.8.3 PUT /subscriptions/{subscription_id}

Update the subscription for events

5.4.8.3.1 Request : PUT /api/v1/subscriptions/{subscription_id}

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret

2 Body
- 2.1 name
- 2.2 callbackUri

5.4.8.3.2 Example

Headers

```
{
	"auth": {
		" client_id": "123456789",
		" client_secret": "lge"
	}
}
```

Body

```
{
	“name”: “Pay Channel”,
	"callbackUri ": " https://procentric.lge.com:60080/api/v2/events/channels"
}
```

5.4.8.3.3 Responses

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Content-Type: application/json
- 1.2 Body
  - 1.2.1 status
  - 1.2.2 data
    - 1.2.2.1 id
    - 1.2.2.2 name
    - 1.2.2.3 created
    - 1.2.2.4 updated
    - 1.2.2.5 callbackUri

2 401 (Unauthorized)

3 404 (Not Found)

5.4.8.3.4 Example

```
{
	“status”: “success”,
	“data”: {
		"id":27,
		"created": "2017-05-26T00:30:08.138Z",
		"updated": "2017-05-27T00:30:08.138Z",
		"callbackUri": "https://localhost:60080/api/v2/events/channels"
	}
}
```

5.4.8.4 DELETE /subscriptions/{subscription_id}

Delete the subscription which is registered in pay channel interface server

5.4.8.4.1 Request : DELETE /api/v1/subscriptions/{subscription_id}

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret

5.4.8.4.2 Example

```
Headers
{
	"auth": {
		" client_id": "123456789",
		" client_secret": "lge"
	}
}
```

5.4.8.4.3 Responses

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Content-Type: application/json
- 1.2 Body
  - 1.2.1 status

2 401 (Unauthorized)

3 404 (Not Found)

5.4.8.4.4 Example

```
{
	“status” : “success”,
	“data”: 1
}
```

5.4.9 Events Nofitication to PCS

Events refers to a time-ordered collection of events. The events has collections of all event type.

HEADER

Attribute	Type	Description
token	String	Token value issued from PCS via POST/PUT subscription

BODY

Attribute	Type	Description
id	Integer	Event identifier
created	String	Created date of the event
event objects	object	Room number

5.4.9.1 POST /events/channels

Retrieve all events. This api is provided by PCS to call event notification such as changing purchased channel list. Subscribe to the pay channel interface server for receiving the events. This api would be callbackUrl's value as the parameter of subscriptions.

5.4.9.1.1 Requests

1 Headers
- 1.1 Cache-Control: no-cache
- 1.2 token

2 Body
- 2.1 id : event ID
- 2.2 created time
- 2.3 events – array
  - 2.3.1 id : sequence ID
  - 2.3.2 type : "channels"
  - 2.3.3 roomId : room ID
  - 2.3.4 tvRoomNumber : tv ID, In case of 2 TVs in a room, it can be specified a specific TV.
  - 2.3.5 paidChannels – array
    - 2.3.5.1 logicalChannelNumber : unique channel ID
    - 2.3.5.2 expireDate : purchased period, timestamp data type
    - 2.3.5.3 metadata : extra channel's information

5.4.9.1.2 Example

HEADER

```
{
    Authorization: Bearer WVFJMzlMcmVBWjJmSU1EdHJjeU13bU9OVFZHQmNFc2FqdemDhjS8bM8Ghq4EgdAxPtafPQ
}
```

BODY

```
{
        "events": [{
                "roomId": "201",,
                "type": "channels",
                "paidChannels": [{
                        "logicalChannelNumber": 6,
                        "type": "IP",
                        "streamType": 16,
                        "encrypted": "false",
                        "ipAddress": "236.0.0.4",
                        "ipPort": 1234,
                        "programNumber": 4,
                        "PCR_PID": 0,
```

```
                        "videoPID": 0,
                        "videoType": -1,
                        "audioPID": 0,
                        "audioType": -1,
                        "logoFileName": "Much_Music_Logo.png",
                        "languages": ["en_US", "ja_JP"],
                        "channelMapListing": "0",
                        "channelLabel": "Servigio23",
                        "channelCategory": "General",
                        "siteId": "0219127",
                        "_id": "X5z1fFbAONRvZTio",
                        "expireDate": "2018-07-31 18:15:05"
                    }]
            }]
}
```


5.5 API Client – Room Control

5.5.1 Room Control Server Configuration

An admin should input the room control server information at PCS's Admin Web Client. The "room control" means that a guest in a room can control devices thru LG hotel TV. The menu is Setting → External Server in the admin client page. The input areas are changed as choosing the server name. A detailed description of the server settings can be found in the Check API(5.5.4 check) for each server.

Explanation about the input area:

- Use Button : Select ON or OFF for the server connection. If selection is On, PCS checks the connection to the external server.
- RC Type : Choose room control server's type. Usually the type is open API. But some hotel chains want to put their interface modules in PCS, so it could be company's name or hotel name.
- Type : Using SSL or not. It depends on external server security.
- Host :The External Interface Server IP address.
- Port : The External Interface Server port number.
- Prefix : The default prefix(5.5.3 Prefix) can be changed when PCD calls GRMS(Guest Room Management Server)'s API.
- Noti. Event : Sending events such as check in/out to external server by PMS server. If GRMS(Guest Room Management Server) has PMS capabilities, the GRMS does not need to be notified when a guest is checking in / out. PCD can deliver the check in/out events to the GRMS(Guest Room Management Server).
- Authorization Section : This section is for authentication or to distinguish hotels. See the 4.2 authentication section. "client_id", "client_secret" keys are mendatory. "client_id" means hotel's unique site ID and "client_secret" is needed to get token for calling PCD's api.
- Message Broker section : To control devices, some interface server uses message broker, so this section is to know message broker server's information. The server can be located in PCS or outside of PCD. In the PCD 3.5, using message broker server to communicate with GRMS(Guest Room Management Server) is not supported.

When an admin clicks the "SAVE" button, PCD calls "POST /check" API to verify input values. Refer 5.5.4 Check Section.

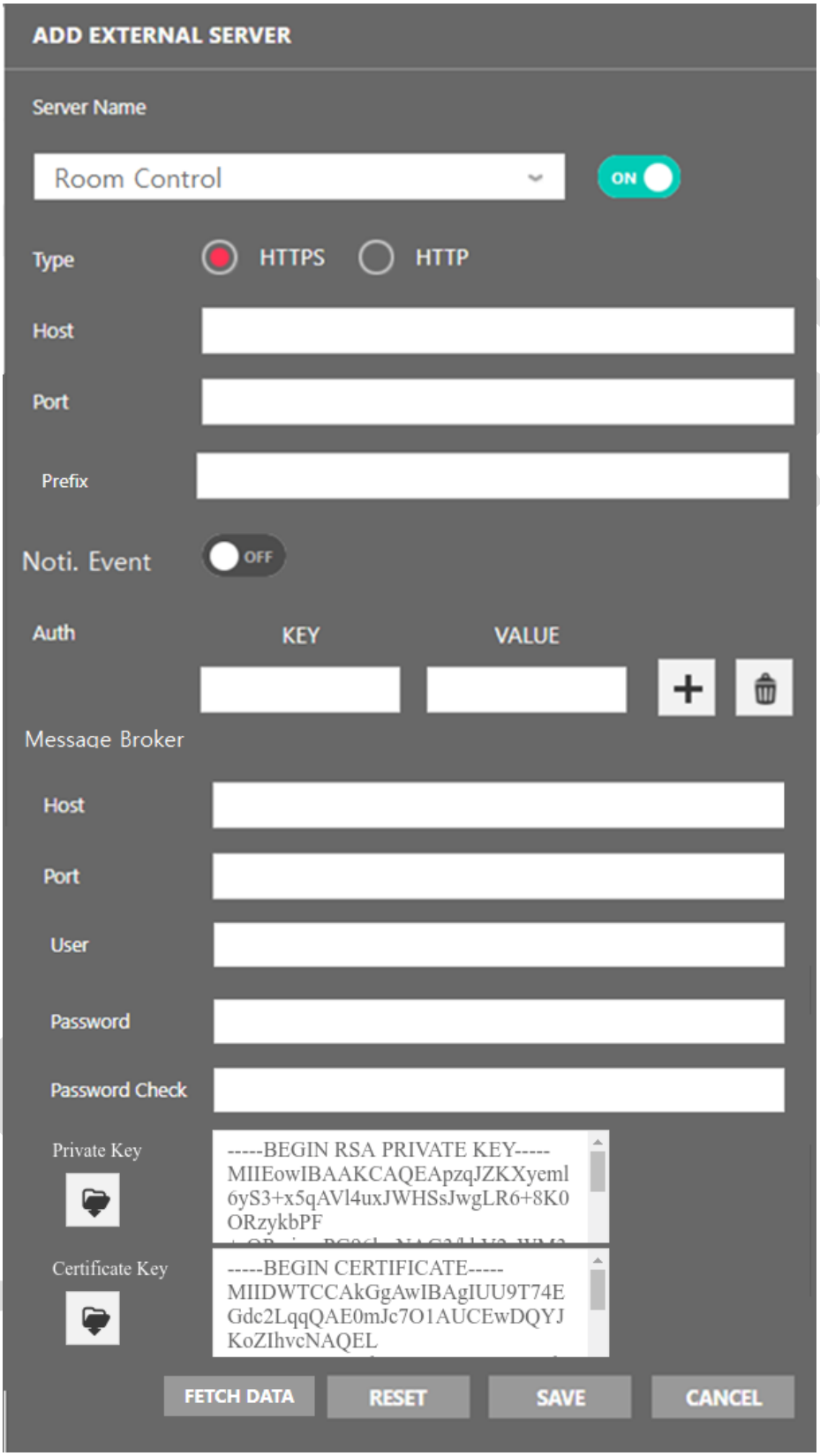

Figure 8

5.5.2 Scenarios to communicate with room control server

The basic sceinarios are below. Refer below sequence diagram (Figure 9)

- Room Control Server configuration : After putting the server information in the Admin Client, when an admin clicks “SAVE” button, PCD calls “POST /check” API for verification. (Section 5.5.4)
- Device Settings : When an admin clicks “FETCH DATA” button, PCD calls 3 kinds of API for initiation.
  - “GET /groups” : This api is to get the virtual room’s information such as Deluxe room type or Suite room type. Each rooms has their own room type and fixed devices. (Section 5.5.6.1).
  - “GET /rooms/{roomId}” : Getting things list in the room. (Section 5.5.6.2).
  - “POST /subscriptions” : To receive events from GRMS, PCD subscribes to the GRMS or Message broker server. (Section 5.5.10)
- Device Control : PCD send a command to change the device’s status. (Section 5.5.7.1).
- Event Notification : When the device’s status is changed, GRMS sends the event to PCD by calling API. Using Message broker server is not supported in the PCD 3.5. (Section 5.5.11.2)

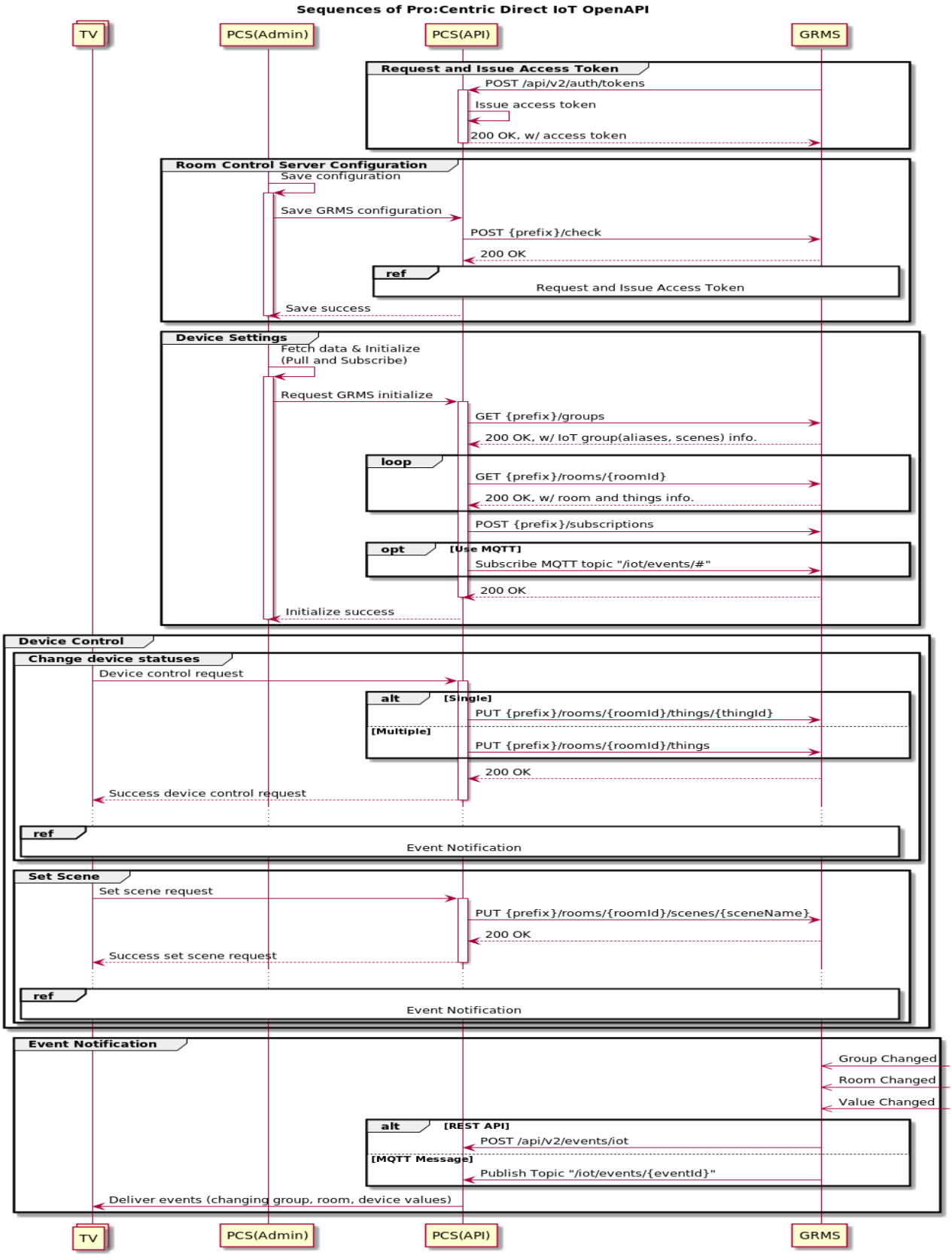

Figure 9

5.5.3 Prefix

It is defined prefix to call Room Control APIs.

/api/iot/v2

5.5.4 Check

PCD calls “POST /check” api to confirm that an admin input the information correctly. If the information is correct, the room control server simply returns "200 OK".

5.5.4.1 Requests : POST /api/iot/v2/check

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret
  - 1.2.3 … (All values in the auth section at PCD’s External server setting page)

2 Body
- 2.1 name : “ROOM CONTROL”
- 2.2 host : server’s IP
- 2.3 port : server’s Port
- 2.4 isSSL : use SSL for security
- 2.5 isNotifyEvent : need or not to be notified when a guest is checking in / out
- 2.6 prefix : set prefix when PCD calls GRMS’s APIs
- 2.7 auth : need to check authorization
- 2.8 msgBroker : message broker server information

5.5.4.2 Example

```
Headers
{
   "auth": {
      "client_id": "123456789",
      "client_secret": "lge"
   }
}
Body
{
   "name": "RoomControl",
   "rcType": " OpenAPI",                    // OpenAPI, LG_RMS, IoT
   "host": "1.2.3.45",
   "port": 8888,
   "isSSL": 0,
   "isNotifyEvent": false,
   "prefix": "/api/v1",
   "auth": {
      "client_id": "123456789",
```

```
        "client_secret": "lge"
    },
    "msgBroker": {
    }
}
```

5.5.5 Device Type, Component ID definition

Each things are related with device types and component ID. And it also rerates the value types.

Attribute Name	Values
Device Type	Light, Dimming, Curtain, Thermostat, DoorLock, BinarySensor, MutiLevelSensor
Component ID	Dimming, OnOff, OpenClose, TargetTemperature, CurrentTemperature, TargetTemperature_F, CurrentTemperature_F, Humidity, Color, ColorTemperature, Level, Speed, Mode
Component Type	ComponentOnOff, ComponentDimmingLight, ComponentDimming, ComponentColor, ComponentFanState, ComponentState, ComponentHumidityState, ComponentVoid, ComponentIntState, ComponentStringState,
Value Type	String, Boolean, Variables, Double, Float, Int8, Int16, Int32, Int64, UInt8, UInt16, UInt32, UInt64, Void

The relation between Device Type and Component is refer the below tables.

Device / Component	Light	Dimming	Thermostat	Curtain	DoorLock	Binary Sensor	MultiLevel Sensor
OnOff	Boolean	Boolean	Boolean				
Dimming		Int8 0 ~ 100					
OpenClose				Boolean	Boolean	Boolean	
TargetTemperature			Double				
CurrentTemperature			Double				
TargetTemperature_F			Double				
CurrentTemperature_F			Double				
Humidity			Double				
Color	String	String					
ColorTemperature	String	String					
Level							"Int8" 0 ~ 100
Speed			INVALID, OFF, LOW, MEDIUM, HIGH				
Mode			COOLING, HEATING, DRY, FAN, AUTO				

5.5.6 Device Settings

When an admin clicks “Fetch Data” button on External Server settings page, PCD calls two kinds of APIs.
First PCD calls “GET /groups” API to get abstracted room information. Refers 2.1.
Second, PCD calls “GET /rooms” APIs.

5.5.6.1 Abstracted room information

The PCD creates an abstract room that can be identified the room's device settings. This way does not need to set all the rooms, and it only sets the types of several rooms such as deluxe room or suites room. In this step, an admin also defines several scenes to control multi device or to follow some scenarios such as wake up scene.

5.5.6.1.1 Groups

The abstracted room is called “Groups” in the PCD. There are several APIs to synch defined abstracted rooms between GRMS(Guest Room Management System) and PCD. If the abstracted room information is changed, GRMS have to notify thru event API.(Refer)

5.5.6.1.1.1 GET /groups

When an admin clicks “Fetch Data” button, PCD calls “GET /groups” api to get logical information of abstracted rooms. (PCD → GRMS)

5.5.6.1.1.2 Requests : GET /api/iot/v2/groups

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret
  - 1.2.3 … (All values in the auth section at PCD’s External server setting page)

5.5.6.1.1.3 Responses

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Content-Type: application/json
- 1.2 Body
  - 1.2.1 status
  - 1.2.2 data (array)
    - 1.2.2.1 name
    - 1.2.2.2 description
    - 1.2.2.3 aliases (array)
      - 1.2.2.3.1 name
      - 1.2.2.3.2 type : Dimming, Light, Curtain, Thermostat, DoorLock, BinarySensor, MultiLevelSensor
    - 1.2.2.4 scenes (array)
      - 1.2.2.4.1 name
      - 1.2.2.4.2 icon
      - 1.2.2.4.3 event
      - 1.2.2.4.4 apply : is Enable? Or Disable?
      - 1.2.2.4.5 intents (array) : optional
      - 1.2.2.4.6 description
      - 1.2.2.4.7 actions (array)
        - 1.2.2.4.7.1 thingAliasName
        - 1.2.2.4.7.2 thingComponentList (array)
          - 1.2.2.4.7.2.1 componentId
          - 1.2.2.4.7.2.2 componentValue
            - 1.2.2.4.7.2.2.1 valueType
            - 1.2.2.4.7.2.2.2 value
            - 1.2.2.4.7.2.2.3 additionalValue

2 401 (Unauthorized)

3 404 (Not Found)

5.5.6.1.1.4 Example


```
{
    "status" : "success",
    "data": [{
    "name": "Deluxe",
    "description": "Lighting(1)",
    "aliases": [
      {
        "name": "DESK_LAMP",
        "type": "Lighting"
      }
    ],
    "scenes": [
      {
        "name": "checkin",
        "icon": null,
        "event": "CHECK_IN",
        "apply": 1,
        "intents": "[]",
        "description": "",
        "actions": [
          {
            "thingAliasName": "DESK_LAMP",
            "thingComponentList": [{
                    "componentId": "Dimming",
                    "componentValue": {
                     "additionalValue": "",
                     "value": "20",
                     "valueType": "Int8"
                  }
            }]
          }
        ]
      },
      {
        "name": "checkout",
        "icon": null,
        "event": "CHECK_OUT",
        "apply": 1,
        "description": "",
        "actions": [
          {
            "thingAliasName": "DESK_LAMP",
            "thingComponentList": [{
                  "componentId": "OnOff",
                  "componentValue": {
                  "additionalValue": "",
                  "value": "0",
                  "valueType": "Boolean"
                }
            }]
          }
        ]
      },
```

```
{
   "name": "normal",
   "icon": null,
   "event": "",
   "apply": 1,
   "description": "",
   "actions": ": [
      {
         "thingAliasName": "DESK_LAMP",
         "thingComponentList": [{
                  "componentId": "OnOff",
                  "componentValue": {
                   "additionalValue": "",
                   "value": "0",
                   "valueType": "Boolean"
                }
         }]
      }
   ]
},
{
   "name": "sleep",
   "icon": null,
   "event": "",
   "apply": 1,
   "description": "",
   "actions": ": [
      {
         "thingAliasName": "DESK_LAMP",
         "thingComponentList": [{
                  "componentId": "OnOff",
                  "componentValue": {
                   "additionalValue": "",
                   "value": "0",
                   "valueType": "Boolean"
                }
         }]
      }
   ]
},
{
   "name": "off",
   "icon": null,
   "event": "",
   "apply": 1,
   "description": "",
   "actions": [
      {
         "thingAliasName": "DESK_LAMP",
         "thingComponentList": [{
                  "componentId": "Dimming",
                  "componentValue": {
                   "additionalValue": "",
                   "value": "20",
```

```
                "valueType": "Int8"
              }
            }]
          }
        ]
      }
    ]
  }]
}
```

5.5.6.2 Device information in the room

The PCD has to know the real device ID in the room. To know the device information in the room, PCD calls "GET /rooms" API.

5.5.6.2.1 GET /rooms/{roomId}

When an admin clicks "Fetch Data" button, PCD calls "GET /rooms/{roomId}" api to get device information in the rooms. (PCD → GRMS). The target value in the below response attributes is TV decides to call TV's HCAP API directly, or call PCD's API to control things.

5.5.6.2.1.1 Requests : GET /api/iot/v2/rooms/{roomId}

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret
  - 1.2.3 … (All values in the auth section at PCD's External server setting page)

5.5.6.2.1.2 Responses

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Content-Type: application/json
- 1.2 Body
  - 1.2.1 status
  - 1.2.2 data
    - 1.2.2.1 roomId
    - 1.2.2.2 iotGroupName
    - 1.2.2.3 things (array)
      - 1.2.2.3.1 id
      - 1.2.2.3.2 name
      - 1.2.2.3.3 description : optional, metadata
      - 1.2.2.3.4 type : Dimming, Light, Curtain, Thermostat, DoorLock, BinarySensor, MultiLevelSensor
      - 1.2.2.3.5 target : things is connected directly by using LG dongle or not. (LG | OpenAPI)
      - 1.2.2.3.6 thingAliasName
      - 1.2.2.3.7 thingComponentList (array)
        - 1.2.2.3.7.1 componentGettable
        - 1.2.2.3.7.2 componentSettable
        - 1.2.2.3.7.3 componentId : Dimming, Off, On, OnOff
        - 1.2.2.3.7.4 componentType : refer the section 5.2 Component ID & Device Type
        - 1.2.2.3.7.5 componentValueType
        - 1.2.2.3.7.6 parentThingUid
        - 1.2.2.3.7.7 componentValue

1.2.2.3.7.7.1 value
1.2.2.3.7.7.2 additionalValue : custom additional value as a string
1.2.2.3.7.7.3 valueType : "String", "Boolean", "Variables", "Double", "Float", "Int8", "Int16", "Int32", "Int64", "UInt8", "UInt16", "UInt32", "UInt64"

2 401 (Unauthorized)
3 404 (Not Found)

5.5.6.2.1.3 Example


```
{
        "status": "success",
        "data": {
                "roomId": "201",
                "iotGroupName": "Deluxe",
                "things": [{
                        "id": "Zigbee:01E6C0000000B8D1F0",
                        "name": "BR30 RGBW",
                        "description": {
                                "hardwareVersion": "48",
                                "label":
"{'clusters':[0003,0004,0005,0006,0008,0B05,0702,0000],'dtype':257,'nwkId':91357,'ndesc':'02408C00'}",
                                "manufacturer": "LG Electronics",
                                "modelName": "B0940EB0Z01",
                                "nickname": "test",
                                "registrationDate": "Tue Jul 24 22:37:09 2018\n",
                                "softwareVersion": "03"
                        },
                        "type": "ThingDimmableLight",
                           "target": "OpenAPI",
                        "thingAliasName": "DESK_LAMP",
                        "thingComponentList": [{
                                "componentGettable": true,
                                "componentId": "Dimming",
                                "componentSettable": true,
                                "componentType": "ComponentDimming",
                                "componentValue": {
                                        "additionalValue": "",
                                        "value": "20",
                                        "valueType": "Int8"
                                },
                                "componentValueType": "Double",
                                "parentThingUid": " Zigbee:01E6C0000000B8D1F0"
                        }, {
                                "componentGettable": true,
                                "componentId": "OnOff",
                                "componentSettable": false,
                                "componentType": "ComponentOnOff",
                                "componentValue": {
                                        "additionalValue": "",
                                        "value": "1",
                                        "valueType": "Boolean"
                                },
                                "componentValueType": "Boolean",
                                "parentThingUid": "Zigbee:01DF84790C006F0D00"
                        }]
```

```
        }, {
                "id": "Zigbee:00:17:88:01:02:93:6c:0a-0b",
                "name": "LCT010",
                "description": "",
                "type": "Light",
                   "target": "OpenAPI",
                "thingAliasName": "DESK_LAMP_L",
                "thingComponentList": [{
                        "componentGettable": true,
                        "componentId": "OnOff",
                        "componentSettable": false,
                        "componentType": "ComponentOnOff",
                        "componentValue": {
                                "additionalValue": "",
                                "value": "1",
                                "valueType": "Boolean"
                        },
                        "componentValueType": "Boolean",
                        "parentThingUid": "Zigbee:01DF84790C006F0D00"
                }]
        }]
    }
}
```


5.5.6.3 Changing device information

When the device information is changed, GRMS has to send the information to PCD by calling event API or Message broker server. Refer the 5.6 Calling event API section.

5.5.7 Device Controlling

There are 3 kinds of API to control things. (1 things control, multi things control, scene control)
The PCD calls an API to control devices such as turn on/off the light or change a scene. (PCD → GRMS)

5.5.7.1 PUT /rooms/{roomId}/things/{thingsId}

When a guest changes the things status, PCD calls “PUT /things” API to control 1 things. (PCD → GRMS).

5.5.7.1.1 Requests : PUT /api/iot/v2/rooms/{roomId}/things/{thingsId}

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret
  - 1.2.3 … (All values in the auth section at PCD’s External server setting page)

2 Body
- 2.1 componentId : refer the section 5.2 Component ID & Device Type
- 2.2 componentValue (Object)
  - 2.2.1 value
  - 2.2.2 additionalValue
  - 2.2.3 valueType

5.5.7.1.2 Example (Request)

```
{
  "componentId": "Dimming",
  "componentValue": {
    "valueType": "Double",
    "value": "1",
    "additionalValue": ""
  }
}
```

5.5.7.1.3 Responses

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Content-Type: application/json
- 1.2 Body
  - 1.2.1 requestId : unique Id for the request
  - 1.2.2 result : Boolean, the result for the request

2 401 (Unauthorized)

3 404 (Not Found)

5.5.7.1.4 Example (Response)

```
{
  "requestId": "13",
  "result": true
}
```

5.5.7.2 PUT /rooms/{roomId}/things

When a guest changes the things status or GRMS doesn't support SCENE mode, PCD calls "PUT /things" API to control multi things. (PCD → GRMS).

5.5.7.2.1 Requests : PUT /api/iot/v2/rooms/{roomId}/things

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret
  - 1.2.3 … (All values in the auth section at PCD's External server setting page)

2 Body (array)
- 2.1 Id : things ID
- 2.2 type : Dimming, Light, Curtain, Thermostat, DoorLock, BinarySensor, MultiLevelSensor
- 2.3 toValues (array) : mandatory
  - 2.3.1 componentId : refer the section 5.2 Component ID & Device Type
  - 2.3.2 componentValue (Object)
    - 2.3.2.1 value
    - 2.3.2.2 additionalValue
    - 2.3.2.3 valueType

5.5.7.2.2 Example (Request)

```
[
    {
        "id" : "Zigbee:01C8BAFB02006F0D00",
        "type": "Light",
```

```
            "toValues" : [{
               "componentId": "Dimming",
               "componentValue": {
                  "valueType": "Double",
                  "value": "1",
                  "additionalValue": ""
               }
            }]
      },
      {
            "id" : "Zigbee:01C8BAFB02006F0D02",
            "type": "Dimming",
            "toValues" : [{
               "componentId": "Dimming",
               "componentValue": {
                  "valueType": "Double",
                  "value": "1",
                  "additionalValue": ""
               }
            }, {
               "componentId": "OnOff",
               "componentValue": {
                  "valueType": "Boolean",
                  "value": "1",
                  "additionalValue": ""
               }
            }]
      }
]
```

5.5.7.2.3 Responses

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Content-Type: application/json
- 1.2 Body
  - 1.2.1 requestId : unique Id for the request
  - 1.2.2 result : Boolean, the result for the request

2 401 (Unauthorized)

3 404 (Not Found)

5.5.7.2.4 Example (Response)

```
{
   "requestId": "13",
   "result": true
}
```

5.5.7.3 PUT /rooms/{roomId}/scenes/{sceneName}

PCD delivers scene command to GRMS, then GRMS changes things in the scene action properties. When the status of things is changed, GRMS sends changing things status event to PCD
(PCD → GRMS).

5.5.7.3.1 Requests : PUT /api/iot/v2/rooms/{roomId}/scenes/{sceneName}

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret
  - 1.2.3 … (All values in the auth section at PCD's External server setting page)

5.5.7.3.2 Responses

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Content-Type: application/json
- 1.2 Body
  - 1.2.1 requestId : unique Id for the request
  - 1.2.2 result : Boolean, the result for the request

2 401 (Unauthorized)

3 404 (Not Found)

5.5.7.3.3 Example (Response)

```
{
  "requestId": "13",
  "result": true
}
```

5.5.8 Device Monitoring

When a guest wants to change the device's status on the TV or APP, PCD can deliver the command to GRMS.
And to show the device's status on the TV, PCD collects the device's status thru event API or messages from message broker (MQTT). When the device's status is changed, GRMS notify the device's status to PCD by calling event API or sending messages thru Message Broker server. Refer the 5.6.1. Event Notification to PCD section.

5.5.9 Receiving check in / out events

The room control server can get check in / out events from PCD if the notification events option is checked in the external server configuration page at Admin Client. When PCD receives the events from PMS, PCD notifies the events to the room control server. The room control server provides below APIs to get the events from PCD.

5.5.9.1 POST /rooms/{room_id}

Notify new rooms information when a guest check in. some values can be omitted for personal security reason.

5.5.9.1.1 Requests : POST /api/v1/rooms/{room_id}

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret

2 Body
- 2.1 id
- 2.2 groupCode
- 2.3 deploymentId

2.4 tvs
2.5 guests

5.5.9.1.2 Example

Headers

```
{
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        }
}
```

Body


```
{
  "id": "201",          // room ID
  "groupCode": "0",
  "deploymentId": 1,
  "tvs": [{
    "tvRoomNumber": "201_a",   // tv ID
    "xait": "ver1",
    "serial": "123SERIAL00A",
    "model": "POST-5500-MAN",
    "platformVersion": "01.02.30.04",
    "micomVersion": "m1.2.3.4",
    "ip": "127.0.0.1",
    "mac": "AB:CD:EF:GH",
    "powerMode": "on",
    "network": "connect",
    "insertAt": "2018-01-25 04:19:03",
    "updateAt": "2018-03-18 14:09:51",
    "roomId": "201",
    "epgFetchDate": "2017-01-test",
    "projectVersion": "1.0",
    "updateStat": "complete"
  }],
  "guests": [{
    "id": "201-1",        // guest ID
    "language": "en",
    "email": "bbaggins@gmail.com",
    "phone": "",
    "vipStatus": "1",
    "payment": "cash",
    "checkin": "2017-05-07 20:10:39",
    "title": null,
    "fullName": "Bilbo  o'Baggins",
    "lastName": "o'Baggins",
    "firstName": "Bilbo",
    "middleName": "",
    "prefix": "",
    "suffix": "",
    "affinityProgram": "",
    "affinityMember": "6547983218919",
    "affinityStatus": "GOLD",
    "groupNumber": "12382734",
    "groupCode": "Baggin's Board Meeting",
```

```
    "insertAt": "2018-03-27 23:49:30",
    "updateAt": "2018-03-27 23:49:30",
    "roomId": "201"
  }]
}
```


5.5.9.1.3 Responses

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Content-Type: application/json
- 1.2 Body
  - 1.2.1 status

2 401 (Unauthorized)

3 404 (Not Found)

5.5.9.1.4 Example


```
{
        "status": “success”,
}
```


5.5.9.2 DELETE /rooms/{room_id}

When PCD receives the check out events, PCD notifies the event to room control server. It can create a new scenario if a guest checked out, such as turning off all devices.

5.5.9.2.1 Request : DELETE /api/v1/rooms/{room_id}

1. Headers
   - 1.1 auth
     - 1.1.1 client_id
     - 1.1.2 client_secret

5.5.9.2.2 Example

Headers


```
{
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        }
}
```


5.5.9.2.3 Responses

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Content-Type: application/json
- 1.2 Body
  - 1.2.1 status

2 401 (Unauthorized)

3 404 (Not Found)

5.5.9.2.4 Example

```
{
        “status” : “success”,
}
```

5.5.10 Subscriptions

Subscribe to the room control server for receiving the events such as turned on light. If the message broker configuration is set on the external server setting page, PCD subscribes to the message broker for getting event messages.

Attribute	Type	Description
name	String	Subscriber name
callbackUri	String	Subscribe to the External Server to receive events

5.5.10.1 POST /subscriptions

The PCS’s load will be massive if receiving all device’s events in the hotel. So PCD only subscribes to the room when a guest is viewing the device control widget on the TV to get changing the device’s statuses.

5.5.10.1.1 Requests : POST /api/v1/subscriptions

1 Headers
- 1.1 Content-Type: application/json
- 1.2 auth
  - 1.2.1 client_id
  - 1.2.2 client_secret

2 Body
- 2.1.1 name
- 2.1.2 callbackUri

5.5.10.1.2 Example

Headers

```
{
        "auth": {
                " client_id": "123456789",
                " client_secret": "lge"
        }
}
```

Body

```
{
        “name”: “Room Control”,
        "callbackUri ": " https://procentric.lge.com:60080/api/v2/events/devices"
}
```

5.5.10.1.3 Responses

1 200 (OK)
- 1.1 Headers
  - 1.1.1 Content-Type: application/json
- 1.2 Body
  - 1.2.1 status
  - 1.2.2 data
    - 1.2.2.1 id

1.2.2.2 created
1.2.2.3 name
1.2.2.4 callbackUri

2 401 (Unauthorized)
3 404 (Not Found)

5.5.10.1.4 Example


```
{
    “status” : “success”,
    "data": {
        "id": 1,
        "created": "2018-04-03T07:49:07+00:00",
        "name": "Room Control",
        "callbackUri": "https://procentric.lge.com:60080/api/v2/events/devices"
    }
}
```


5.5.10.2 DELETE /subscriptions

It stops the subscription when a guest closes the room control widget. There is no longer need to see the state changes.

5.5.10.2.1 Request : DELETE /api/v1/subscriptions

1 Headers
1.1 Content-Type: application/json
1.2 auth
1.2.1 client_id
1.2.2 client_secret

5.5.10.2.2 Example


```
Headers
{
    "auth": {
        " client_id": "123456789",
        " client_secret": "lge"
    }
}
```


5.5.10.2.3 Responses

1 200 (OK)
1.1 Headers
1.1.1 Content-Type: application/json
1.2 Body
1.2.1 status
2 401 (Unauthorized)
3 404 (Not Found)

5.5.10.2.4 Example


```
{
    “status” : “success”,
    “data”: 1
}
```

5.5.11 Events Nofitication to PCS

When the device's information or status is changed, GRMS notify the information to PCD by calling below API or sending messages thru Message Broker server.

5.5.11.1 Calling event API

GRMS has to notify when below information is changed.

- Groups : The name of the device constituting the room is changed or added.
- Rooms : The things is constituted or added.
- Values : The things' status is changed.

5.5.11.1.1 Events

When an admin clicks "Fetch Data" button on Admin Client, PCD tell the callback URL to receive event. This api would be callbackUrl's value as the parameter of subscriptions. The event handing API is "POST /api/v2/events/iot". When GRMS calls this API, GRMS has to put the access token value in the header.
When an admin clicks "Fetch Data" button, PCD gets all device information from GRMS. But sometimes only 1 or a few devices are changed or added, for this case, GRMS sends the changed information to the PCD by calling event API. (GRMS → PCN)

Events refers to a time-ordered collection of events. The events has collections of all event type.
HEADER

Attribute	Type	Description
token	String	Token value issued from PCS via POST/PUT subscription

BODY

Attribute	Type	Description
id	Integer	Event identifier (Unique & Increased ID)
type	String	event types, "groups", "rooms", "values"
method	String	action type : "PUT", "POST", "DELETE"
data	Array	event object : groups, rooms, values

5.5.11.1.1.1 POST /api/v2/events/iot

Retrieve all events. This API is provided by PCS to call event notification such as turned lights on. Subscribe to the room control interface server or message broker for receiving the events. This api would be callbackUrl's value as the parameter of subscriptions.

5.5.11.1.1.1.1 Requests

1 Headers
  1.1 Cache-Control: no-cache
  1.2 Authorization
2 Body
  2.1 Values – array
    2.1.1 id : unique and increased event ID
    2.1.2 type : event types, "groups", "rooms", "values"
    2.1.3 method : action type : "PUT", "POST", "DELETE"
    2.1.4 data (array) : event object : groups, rooms, values

5.5.11.1.1.1.2 Example (Groups is changed)

The “intents” attributes for the voice recognition cannot be changed by GRMS, it will be kept the intents list by PCD when the method value is “PUT”.

HEADER
{
Authorization: Bearer JMzlMcmVBWjJmSU1EdHJjeU13bU9OVFZHQmNFc2FqdemDhjS8bM8Ghq4EgdAxPtafPQ
}
BODY
{

```
"id": "123",
"type": "groups",
 "method": "PUT",
"data": [{
        "name": "Deluxe",                                   // unique ID, mandatory
        "description": "Lighting",
        "aliases": [{
                "name": "DESK_LAMP",
                "type": "Dimming"
        }],
        "scenes": [{
                        "name": "normal",
                        "icon": null,
                        "event": "",
                        "apply": 1,
                        "description": "",
                        "actions": [{
                                "thingAliasName": "DESK_LAMP",
                                "thingComponentList": [{
                                        "componentId": "Dimming",
                                        "componentValue": {
                                                "additionalValue": "",
                                                "value": "20",
                                                "valueType": "Int8"
                                        }
                                }]
                        }]
                },
                {
                        "name": "sleep",
                        "icon": null,
                        "event": "",
                        "apply": 1,
                        "description": "",
                        "actions": [{
                                "thingAliasName": "DESK_LAMP",
                                "thingComponentList": [{
                                   "componentId": "OnOff",
                                   "componentValue": {
                                   "additionalValue": "",
                                   "value": "0",
                                   "valueType": "Boolean"
                                }
```

```
                                        }]
                                }]
                        },
                        {
                                "name": "off",
                                "icon": null,
                                "event": "",
                                "apply": 1,
                                "description": "",
                                "actions": [{
                                        "thingAliasName": "DESK_LAMP",
                                        "thingComponentList": [{
                                           "componentId": "OnOff",
                                           "componentValue": {
                                           "additionalValue": "",
                                           "value": "0",
                                           "valueType": "Boolean"
                                        }
                                }]
                                }]
                        }
                ]
        }]
}
```


5.5.11.1.1.1.3 Example (Rooms is changed)

HEADER


```
{
Authorization: Bearer JMzlMcmVBWjJmSU1EdHJjeU13bU9OVFZHQmNFc2FqdemDhjS8bM8Ghq4EgdAxPtafPQ
}
```


BODY


```
{
        "id": "123",
        "type": "rooms",
        "method": "PUT",
        "data": [{
                "roomId": "201",
                "iotGroupName": "Deluxe",
                "things": [{
                        "id": "Zigbee:01E6C0000000B8D1F0",
                        "name": "BR30 RGBW",
                        "description": {
                                "hardwareVersion": "48",
                                "label":
"{'clusters':[0003,0004,0005,0006,0008,0B05,0702,0000],'dtype':257,'nwkId':91357,'ndesc':'02408C00'}",
                                "manufacturer": "LG Electronics",
                                "modelName": "B0940EB0Z01",
                                "nickname": "DESK_LAMP",
                                "registrationDate": "Tue Jul 24 22:37:09 2018\n",
                                "softwareVersion": "03"
                        },
                        "type": "Dimming",
```

```
                        "thingAliasName": "DESK_LAMP",
                        "thingComponentList": [{
                                "componentGettable": true,
                                "componentId": "Dimming",
                                "componentSettable": true,
                                "componentType": "ComponentDimming",
                                "componentValue": {
                                        "additionalValue": "",
                                        "value": "1.000000",
                                        "valueType": "Double"
                                },
                                "componentValueType": "Double",
                                "parentThingUid": " Zigbee:01E6C0000000B8D1F0"
                        }, {
                                "componentGettable": true,
                                "componentId": "OnOff",
                                "componentSettable": false,
                                "componentType": "ComponentOnOff",
                                "componentValue": {
                                        "additionalValue": "",
                                        "value": "1",
                                        "valueType": "Boolean"
                                },
                                "componentValueType": "Boolean",
                                "parentThingUid": "Zigbee:01E6C0000000B8D1F0"
                        }]
                }, {
                        "id": "Zigbee:01DF84790C006F0D00",
                        "name": "LCT010",
                        "description": "",
                        "type": "Light",
                        "thingAliasName": "DESK_LAMP_L",
                        "thingComponentList": [{
                                "componentGettable": true,
                                "componentId": "OnOff",
                                "componentSettable": false,
                                "componentType": "ComponentOnOff",
                                "componentValue": {
                                        "additionalValue": "",
                                        "value": "1",
                                        "valueType": "Boolean"
                                },
                                "componentValueType": "Boolean",
                                "parentThingUid": "Zigbee:01DF84790C006F0D00"
                        }]
                }]
        }]
}
```


5.5.11.1.1.1.4 Example (Value is changed)

HEADER


```
{
```

```
Authorization: Bearer JMzlMcmVBWjJmSU1EdHJjeU13bU9OVFZHQmNFc2FqdemDhjS8bM8Ghq4EgdAxPtafPQ
}
BODY
{
        "id": "123",
        "type": "values",
        "method": "PUT",
        "data": [{
                        "id": "Zigbee:01C8BAFB02006F0D00",
                        "type": "Light",
                        "thingComponentList": [{
                                "componentId": "Light",
                                "componentValue": {
                                        "valueType": "Double",
                                        "value": "1",
                                        "additionalValue": ""
                                }
                        }]
                },
                {
                        "id": "Zigbee:01C8BAFB02006F0D02",
                        "type": "Dimming",
                        "thingComponentList": [{
                                "componentId": "Dimming",
                                "componentValue": {
                                        "valueType": "Double",
                                        "value": "1",
                                        "additionalValue": ""
                                }
                        }, {
                                "componentId": "OnOff",
                                "componentValue": {
                                        "valueType": "Boolean",
                                        "value": "1",
                                        "additionalValue": ""
                                }
                        }]
                }
        ]
}
```

5.5.11.2 Sending messages

GRMS using Message Broker publishes messages when there is changed information.
PCN will subscribe “/iot/events/#” topic to receive event messages. To be able to send an event, GRMS should publish message to “/iot/events/{event_id}” as topic.

A published message consists of 2 JSON objects.

- context : the metadata of this message
- payload : all information for this message

context

Attribute	Type	Description
messageId	String	Mandatory, Unique message ID

description	String	Optional, Description for this message
hotelId	String	Optional, Unique hotel ID
resource	String	Endpoint URL for action
header	JSON	Header information for the endpoint URL
createAt	Datetime	Mandatory, Message creation time
creator	String	Optional, Message distributor (ex. RMS, GRMS)

payload

Attribute	Type	Description
Id	String	Unique event ID
method	String	Method type (PUT | POST | DELETE)
type	String	Mandatory, event types (groups | rooms | values)
data	JSON	Event contents
parameters	JSON	Parameters for the endpoint URL

5.5.11.2.1 Example


```
{
   "context": {
            "messageId": "dnDgTqW0FczYOc6",
             "description": "value is changed",
             "hotelId": "01923412",
             "resource": "/rooms/{roomId}/things/{thingId}",
             "header": {"client_id":"01923412", "client_secret":"password"},
             "createAt": "2018-03-18 14:09:51",
             "creator": "RMS"
       },
   "payload": {
            "id": "dnDgTqW0FczYOc6",
             "type": "values",
             "method": "PUT",
             "data": {
                        "id": "Zigbee:01C8BAFB02006F0D00",
                        "type": "Light",
                        "thingComponentList": [{
                                   "componentId": "Dimming",
                                   "componentValue": {
                                              "valueType": "Int8",
                                              "value": "80",
                                              "additionalValue": ""
                                   }
                        }, {
                                   "componentId": "OnOff",
                                   "componentValue": {
                                              "valueType": "Boolean",
                                              "value": "1",
                                              "additionalValue": ""
                                   }
                        }]
             },
             "parameters": {}
   }
}
```

5.6 API Client – Casting Service

When a guest is checked-in or checked-out, PCS notify it to casting server and TV. Casting server sends the welcome message and token value to casting device. TV changes into HDMI mode. This function calls Connect My Device.

5.6.1 Prefix

It is defined prefix to call Casting service APIs.

/api/cast

5.6.2 Check in/out Notification

When a guest check in/out, PCS notifies to casting server.

5.6.2.1 GET /rooms/{room_id}

When a guest check in, PCS calls get rooms method to notify.

5.6.2.1.1 Requests : GET /api/cast/rooms/{room_id}

5.6.2.1.2 Responses

1 200 (OK)
2 404 (Unkown room)
3 400 (Server error)
4 500 : (Error)

5.6.2.2 DELETE /rooms/{room_id}

When a guest check out, PCS calls get rooms method to notify.

5.6.2.2.1 Requests : DELETE /api/cast/rooms/{room_id}

5.6.2.2.2 Responses

1 204 (No Content)
2 404 (Unkown room)
3 400 (Room was not checked-in)
4 500 : (Error)

5.7 API Client – TV Control

PCD provides APIs to control TVs remotely. You can control only one TV or deployment grouped TVs.
To use TV control API, You need to set your external server to get authorized token from PCD(4.1 External Server Setting). Then your system can get an access token (refer the section 4.2).

5.7.1 Prefix

It is defined prefix to call TV control service APIs.

/api/v2

5.7.2 Getting TV information from PCD

PCD provides all TVs' information in the hotel. You can get all TVs' information by calling below API

5.7.2.1 GET /status/tv

5.7.2.1.1 Requests : GET /api/v2/status/tv

1 Headers
- 1.1 Cache-Control: no-cache
- 1.2 Authorization

5.7.2.1.2 Responses

1 Body
- 1.1 status : API call's status
- 1.2 data : array for TV list

5.7.2.1.3 Examples

```
{
  "status": "success",
  "data": [
    {
      "tvRoomNumber": "201_a",
      "xait": "ver1",
      "serial": "123SERIAL00A",
      "model": "POST-5500-MAN",
      "platformVersion": "01.02.30.04",
      "micomVersion": "m1.2.3.4",
      "ip": "127.0.0.1",
      "mac": "AB:CD:EF:GH",
      "powerMode": "unknown",
      "network": "disconnect",
      "insertAt": "2019-08-28 05:01:18",
      "updateAt": "2019-09-09 05:10:51",
      "roomId": "201",
      "epgFetchDate": "2017-01-test",
      "projectVersion": "1.0",
      "softApPassword": null,
      "softApMode": 0,
      "signalStrength": null,
      "lastStatusUpdate": "2019-08-29 01:18:01",
      "status": "check-in",
      "updateStat": "complete"
    },
    {
      "tvRoomNumber
      ……
    },
      …….
  ]
}
```

5.7.3 Controlling one TV with one command

TV can be controlled by PCD's API.

5.7.3.1 POST /control/tv/{tv_serial}

5.7.3.1.1 Requests : GET /api/v2/control/tv/{tv_serial}

1 Headers
- 1.1 Cache-Control: no-cache
- 1.2 Authorization

2 Body
- 2.1 command : TV's action (on|off|reboot|hdmi2)

5.7.3.1.2 Examples

Headers
```
{
    "Authorization": "Bearer WVFJMzlMcmVBWjJmSU1EdHJjeU13bU9OVFZHQmNFc2FqdemDhjS8bM8Ghq4EgPQ"
}
```
Body
```
{
    "command": "on"
}
```

5.7.4 Controlling multi TV with multi commands

PCD can control deployment grouped TVs thru open API. The group can be set at the Admin Client.(MANAGER > DEPLOYMENT GROUP). The group can be assigned as Group A, Group B and Group C.

5.7.4.1 POST /control/tv/groups/{deployment_id}

5.7.4.1.1 Requests : GET /api/v2/control/tv/groups/{deployment_id}

3 Headers
- 3.1 Cache-Control: no-cache
- 3.2 Authorization

4 Body
- 4.1 command : json format for multi commands

5.7.4.1.2 Examples

Headers
```
{
    "Authorization": "Bearer WVFJMzlMcmVBWjJmSU1EdHJjeU13bU9OVFZHQmNFc2FqdemDhjS8bM8Ghq4EgPQ"
}
```
Body
```
{
    "command": {"channel":"99", "volume":"30"}          // power (on|off|reboot), channel, volume, input
}
```

6 Summary

6.1 Data exchanging flow

For security reasons, each external servers should get the authorization. Here is how PCS exchanges data with the interface servers.

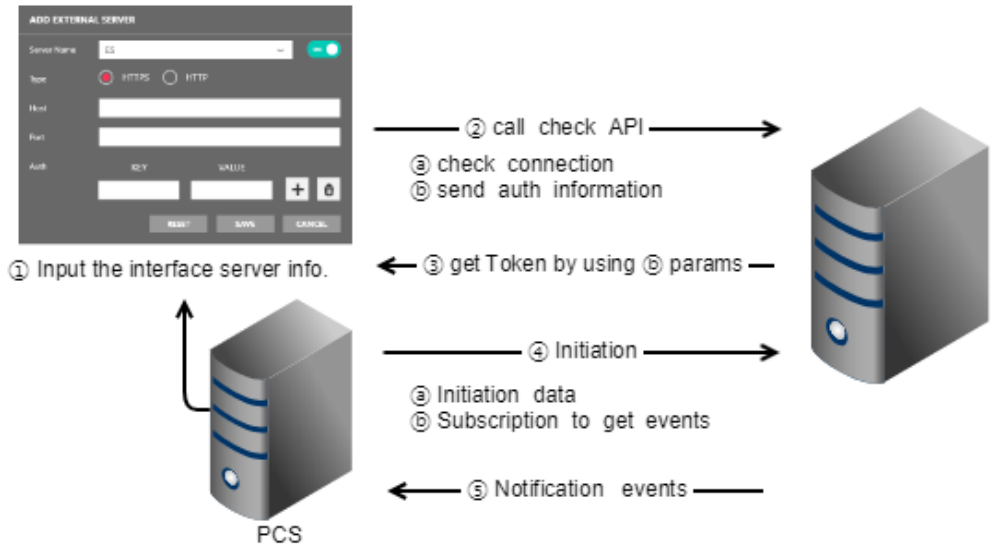

Figure 10

6.2 RESTful API list

6.2.1 PCD interfaces for external servers

Group	Resource	GET(read)	POST(create)	PUT(update)	DELETE
Common	/api/v2/auth/tokens		Get access token		
PMS	/api/v2/events/pms		Receive events from PMS		
QMS	/api/v2/events/qms		Receive events from QMS		
Pay Channel	/api/v2/events/channels		Receive events from Pay channel server		
Room Control	/api/v2/events/devices		Receive events from Room Control server		

6.2.2 PMS interfaces

PCS calls below RESTful APIs to send or receive information from PMS interface server.

6.2.2.1 Prefix

/api/pms/v2

6.2.2.2 Interfaces

Resource	GET(read)	POST(create)	PUT(update)	DELETE
/check		Check connection		
/statuses	Get server status			
/details	Get site info.			
/rooms/{roomId}	Retrieve a room information			
/rooms/{roomId}/folios/{guestId}	retrieve a full `folio` representation			
/rooms/{roomId}/folios/{guestId}/folio-items		create folio item to		

		charge a bill		
/rooms/{roomId}/checkouts		Request a checkout		
/rooms/{roomId}/checkouts/{guestId}		Request a checkout for a specific guest		
/subscriptions		Regist a scubscription		
/subscriptions/{subscriptionId}	Get a subscribed Info		Update a subscription	Delete a subscription

6.2.3 QMS interfaces

PCS calls below RESTful APIs to send or receive information from QMS interface server.

6.2.3.1 Prefix

/api/qms/v2

6.2.3.2 Interfaces

Resource	GET(read)	POST(create)	PUT(update)	DELETE
/check		Check connection		
/services	Get a list of `concierge services` provided by a specific hotel			
/requests/{roomId}		Create new service request	Modify the previous request	
/directories	Get a list of service directories for the hotel			
/rooms/{roomId}		Sending check-in event from PMS		Sending check-out event from PMS
/subscriptions		Regist a scubscription		
/subscriptions/{subscriptionId}	Get a subscribed Info		Update a subscription	Delete a subscription

6.2.4 Pay Channel Interfaces

PCS calls below RESTful APIs to send or receive information from Pay Channel interface server.

6.2.4.1 Prefix

/api/v1

6.2.4.2 Interfaces

Resource	GET(read)	POST(create)	PUT(update)	DELETE
/check		Check connection		
/sites/{siteId}/channels		Send all channel list		
/channels/{channelId}	Get a specific channel info.	Add a new channel	Update channel info.	Delete a channel
/rooms/{roomId}		Add a new room (check in)		Delete a room (check out)
/rooms/{roomId}/channels	Get a current paid channel list			
/subscriptions		Regist a scubscription		
/subscriptions/{subscriptionId}	Get a subscribed Info		Update a subscription	Delete a subscription

6.2.5 Room Control Interfaces

PCS calls below RESTful APIs to send or receive information from Pay Channel interface server.

6.2.5.1 Prefix

/api/iot/v2

6.2.5.2 Interfaces

Resource	GET(read)	POST(create)	PUT(update)	DELETE
/api/iot/v2/check		Check connection		
/groups	Get room's type from GRMS			
/api/iot/v2/rooms/{roomId}	Get device list in a room			
/api/v1/rooms/{roomId}		Send checkin event to GRMS		Send checkout event to GRMS
/api/iot/v2/rooms/{roomId}/things/{thingId}		Set the device's status		
/api/iot/v2/rooms/{roomId}/things		Set the device's configuration		
/api/iot/v2/rooms/{roomId}/scenes/{sceneName}		Set the device's status as putting as buf		
/api/v1/subscriptions		Create a subscription for each room's events		Unregister to subscribe for each room

6.2.6 Casting device interfaces

In case of existing casting server to control casting devices, PCS notify check in/out events to casting server

6.2.6.1 Prefix

/api/cast

6.2.6.2 Intefaces

Resource	GET(read)	POST(create)	PUT(update)	DELETE
/rooms/{roomId}	Notify check in event			Notify check out event

6.2.7 TV control interfaces

In case of existing casting server to control casting devices, PCS notify check in/out events to casting server

6.2.7.1 Prefix

/api/v2

6.2.7.2 Intefaces

Resource	GET(read)	POST(create)	PUT(update)	DELETE
/status/tv	Get all TV information			
/status/tv/{tv_serial}	Get a TV information			
/controls/tv/{tv_serial}		Control a tv with 1 commands		

/controls/tv/groups/{id}		Control multi tvs with multi commands		