User Guide of the OMS_1.0STDx

OMS User Guide

Version 1.1.0

Status: Proposed

Filename	Deployment_directive_and_parameters_OMS_1.0STDx.doc
Document Nr	
Date	January 24, 2014
Author(s)	OMS Team
Information Domain	
Client/Project	OMS 1.x
Data Owner	OMS Team

CH-1033 Cheseaux, Switzerland.
Tel: +41 21 7320311 Fax: +41 21 7320100
www.nagra.com

This document is supplied with an understanding that the notice(s) herein or any other contractual agreement(s) made that instigated the delivery of a hard copy, electronic copy, facsimile or file transfer of this document are strictly observed and maintained.

The information contained in this document is subject to change without notice.

Security Policy of Nagravision SA Kudelski Group

Any recipient of this document, without exception, is subject to a Non-Disclosure Agreement (NDA) and access authorization.

Contents

Preface

Purpose

This document provides an overview and an understanding on how to use and the OMS functionalities.
Audience
This guide is intended for site engineers and users that must use the Module Release OMS_1.0STDx application.

Used Conventions

The used convention for dates in numerical format is YYYYMMDD.
Example: 20090810 is equivalent to August 10th, 2009.

Document History

Version	*Date*	*Author*	*Change*
1.x	2013-12-18	Viraxay Manixab	Initial version for OMS 1.0
1.1	2014-01-24	Viraxay Manixab	Fix example in service provider section

Document Reviewers

Reviewer's Name	**Function**
Frédéric Gorgerat	Test Manager

1. Important preliminary check

1.1 Deployment directive and parameters OMS

Read the document Deployment directive and parameters to understand how to install and launch the OMS software.

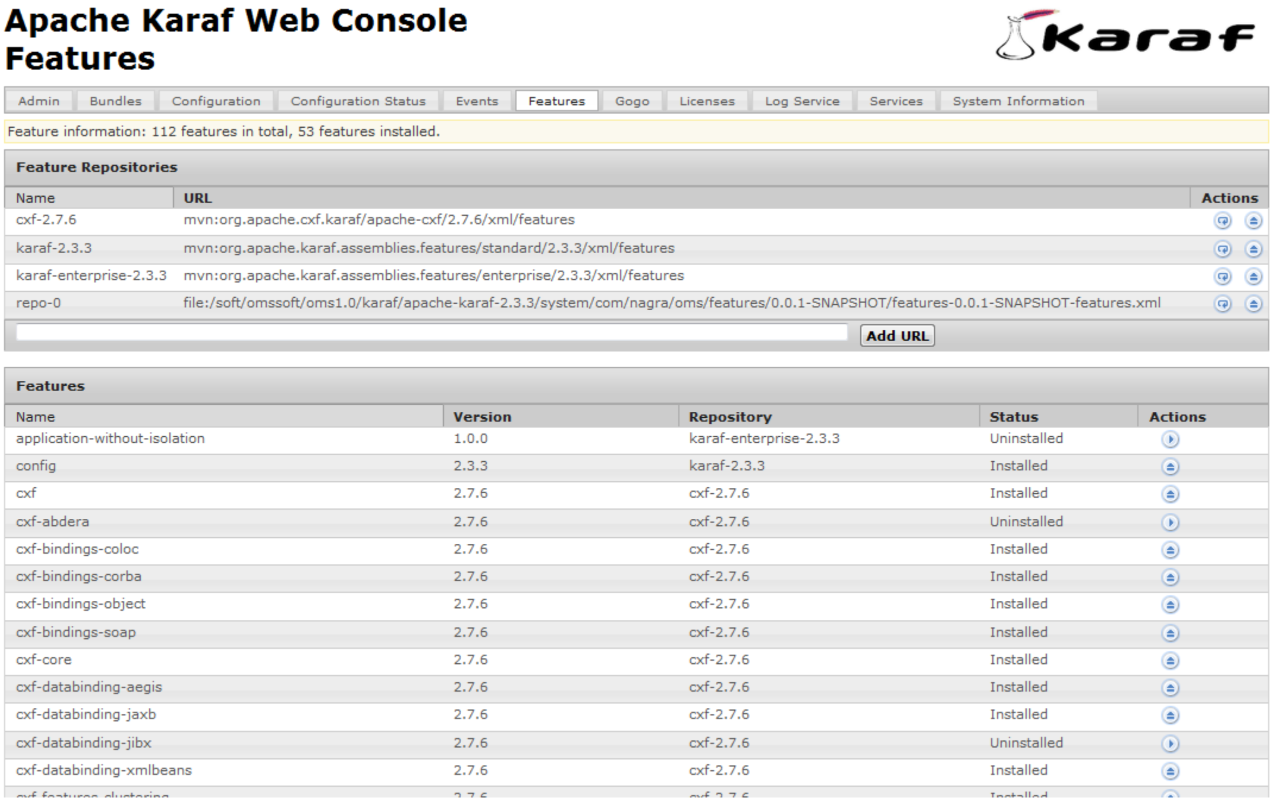
2. OSGi Management of the features

The OSGi management is available on a webpage. This tool is provided by the Apache Karaf Web Console.
The webconsole is reachable at the following address:

```
http://<OSGi-server>:8181/system/console/bundles
```

The webconsole allows:

- to see the logs, with the option of filter by severity
- manage the features (install and uninstall)
- manage the bundles (stop, update, uninstall the bundles and refresh the package imports)
- command console



Feature information: 112 features in total, 53 features installed.

Feature Repositories

Name	URL	Actions
cxf-2.7.6	mvn:org.apache.cxf.karaf/apache-cxf/2.7.6/xml/features	↻ ⏏
karaf-2.3.3	mvn:org.apache.karaf.assemblies.features/standard/2.3.3/xml/features	↻ ⏏
karaf-enterprise-2.3.3	mvn:org.apache.karaf.assemblies.features/enterprise/2.3.3/xml/features	↻ ⏏
repo-0	file:/soft/omssoft/oms1.0/karaf/apache-karaf-2.3.3/system/com/nagra/oms/features/0.0.1-SNAPSHOT/features-0.0.1-SNAPSHOT-features.xml	↻ ⏏

Features

Name	Version	Repository	Status	Actions
application-without-isolation	1.0.0	karaf-enterprise-2.3.3	Uninstalled	▶
config	2.3.3	karaf-2.3.3	Installed	⏏
cxf	2.7.6	cxf-2.7.6	Installed	⏏
cxf-abdera	2.7.6	cxf-2.7.6	Uninstalled	▶
cxf-bindings-coloc	2.7.6	cxf-2.7.6	Installed	⏏
cxf-bindings-corba	2.7.6	cxf-2.7.6	Installed	⏏
cxf-bindings-object	2.7.6	cxf-2.7.6	Installed	⏏
cxf-bindings-soap	2.7.6	cxf-2.7.6	Installed	⏏
cxf-core	2.7.6	cxf-2.7.6	Installed	⏏
cxf-databinding-aegis	2.7.6	cxf-2.7.6	Installed	⏏
cxf-databinding-jaxb	2.7.6	cxf-2.7.6	Installed	⏏
cxf-databinding-jibx	2.7.6	cxf-2.7.6	Uninstalled	▶
cxf-databinding-xmlbeans	2.7.6	cxf-2.7.6	Installed	⏏
cxf-features-clustering	2.7.6	cxf-2.7.6	Installed	⏏

3. Wadl and endpoints

3.1 API Definition

The WADL provides the specification, description to an external system on how to communicate with, in this case, the OMS system.

The endpoint is a terminal address. An endpoint can call and be called. This is the address of where an external system can communicate, in this case, with the OMS system.

Once the OMS started, the WADL and information about endpoints are available on the following address:

http://<OSGi-server>:8181/cxf

Available SOAP services:

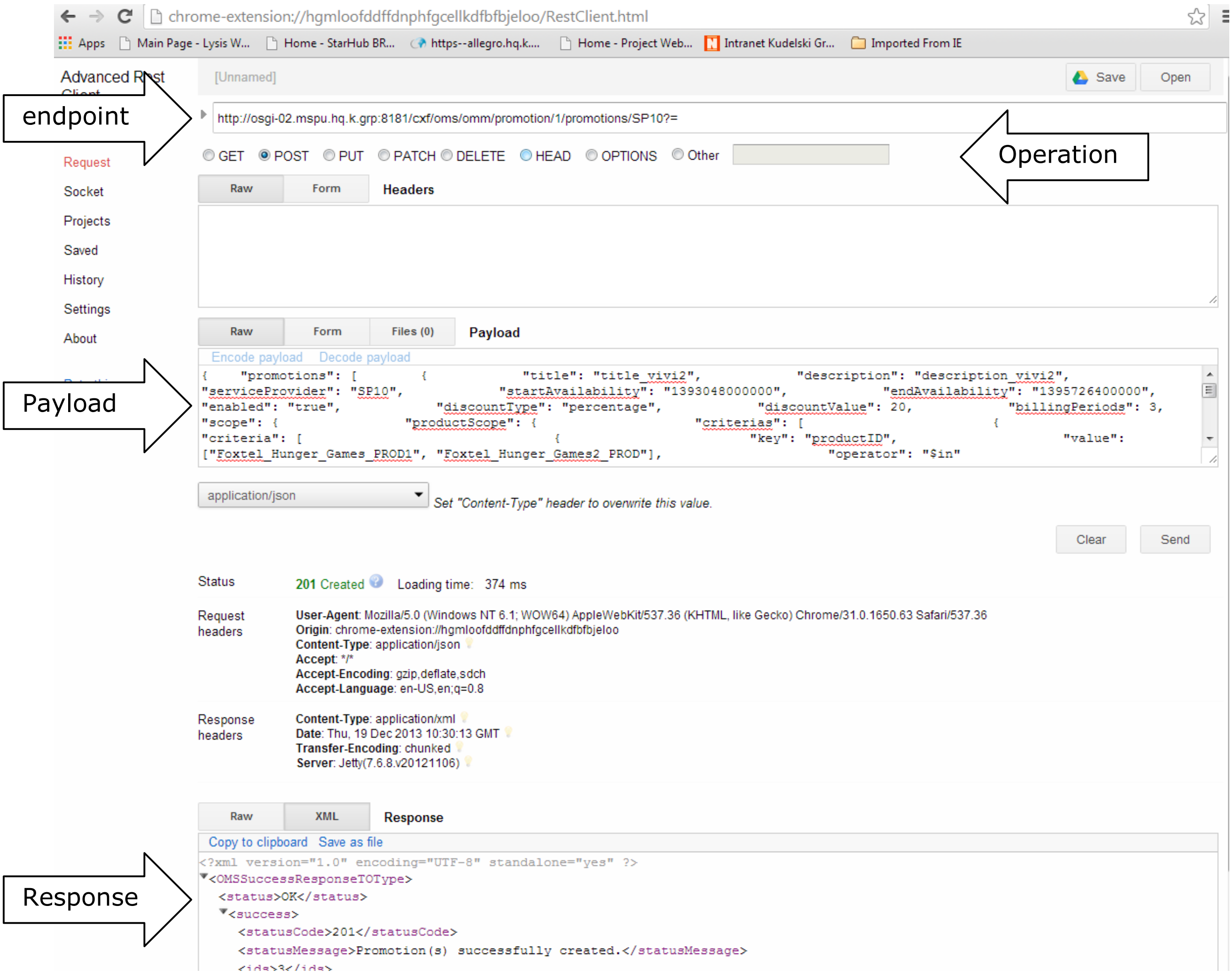
Available RESTful services:

Endpoint address: http://<OSGi-server>:8181/cxf/oms/omm/promotion/1 WADL : http://<OSGi-server>:8181/cxf/oms/omm/promotion/1?_wadl
Endpoint address: http://<OSGi-server>:8181/cxf/oms/omm/notification/1 WADL : http://<OSGi-server>:8181/cxf/oms/omm/notification/1?_wadl
Endpoint address: http://<OSGi-server>:8181/cxf/oms/omm/sp/1 WADL : http://<OSGi-server>:8181/cxf/oms/omm/sp/1?_wadl

3.2 How to operate manually with the endpoints

Multiple tools are available to operate, communicate and interact with OMS software. SOAP UI is one of the tool uses by the test team to test all the features of the OMS.
For basic test, the Advanced REST client tool provided on the Chrome web browser is used.

The interface of the Advanced Rest client is the following:



http://osgi-02.mspu.hq.k.grp:8181/cxf/oms/omm/promotion/1/promotions/SP10?=

```
{     "promotions": [          {               "title": "title_vivi2",            "description": "description_vivi2",
"serviceProvider": "SP10",              "startAvailability": "1393048000000",             "endAvailability": "1395726400000",
"enabled": "true",            "discountType": "percentage",             "discountValue": 20,            "billingPeriods": 3,
"scope": {             "productScope": {                    "criterias": [                    {
"criteria": [                              {                           "key": "productID",                "value":
["Foxtel_Hunger_Games_PROD1", "Foxtel_Hunger_Games2_PROD"],                          "operator": "$in"
```

Set "Content-Type" header to overwrite this value.

Status 201 Created Loading time: 374 ms

Request headers

```
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/31.0.1650.63 Safari/537.36
Origin: chrome-extension://hgmloofddffdnphfgcellkdfbfbjeloo
Content-Type: application/json
Accept: */*
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
```

Response headers

```
Content-Type: application/xml
Date: Thu, 19 Dec 2013 10:30:13 GMT
Transfer-Encoding: chunked
Server: Jetty(7.6.8.v20121106)
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
▼<OMSSuccessResponseTOType>
  <status>OK</status>
 ▼<success>
    <statusCode>201</statusCode>
    <statusMessage>Promotion(s) successfully created.</statusMessage>
    <ids>3</ids>
```

The interface provides:

- A field to enter the endpoint on which we want to do an operation
- Query parameters, corresponding to the query string
- The choice of type of operation: GET, POST, PUT or DELETE
- The Payload, data to communicate to the system
- The response

4. Promotion

The base URL definition for managing the promotions is defined with the endpoint and the WADL. And it is:

```
http://<host>:port/cxf/oms/omm/promotion/<version>/promotions/<SP>?<query_string>
```

- Host is the OSGi host name
- Port of the host, by default is 8181
- Version of the interface. It is 1 for the first release
- SP is the service provider

4.1 Properties

The table below represents the promotion with all its properties:

Property	Type	Description	Sample
id	string	Unique value that fully identifies the Promotion Object	123
title	string	The title of the promotion	Christmas Promo
description	string	The description of the promotion	Everything half the price for Christmas
serviceProvider	string		CMS
startAvailability	dateTime	UTC Date and Time of the Start of the Promotion	2013-10-01T00:00:00Z
endAvailability	dateTime	UTC Date and Time of the End of the Promotion	2013-10-01T00:00:00Z
enabled	boolean	true if the Promotion is active, false if it's deactivated	true/false
discountType	DiscountType	Enumeration representing the possible kind of discount	percentage;finalPrice
discountValue	float	The discount value associated to the discount type: *percentage* discount type: 20.0 means 20.0% *finalPrice* discount type: 20.0 means a final price of 20.0 [currency]	20.0
billingPeriods	long	The number of billing periods associated to this promotion	5
scope	scope	The scope of the promotion.	
priority	int	The priority value associated to the promotion object	2

4.1.1 Scope

The property of a scope is the following:

Property	Type	Description
scope	contentScope / productScope accountScope subscriberProfile	The scope of the promotion, can be either a contentScope object either a productScope object. and can have an account scope and a subscriber profile.

4.1.1.1 contentScope

Property	Type	Description	Sample
productType	string	type of product	"productType": "single"
logicalOperator	string	Logical operator used between each criterias logicaleOperator is not mandatory. If not provided, it will be considered as a AND	"logicalOperator": "$or"
criterias	criteria (maxOccurs="unbounded")	Composed of key, value list and operator	{ "key": " Actor", "value": ["toto"] },

4.1.1.2 productScope

Property	Type	Description	Sample
logicalOperator	string	logical operator used between each criterias logicalOperator is not mandatory. If not provided, it will be considered as a AND Normally the logicalOperator will not be	"logicalOperator": "$or"

		used	

criterias	criteria
	(maxOccurs=
	"unbounded")
		Composed of key, value list and operator. In this case the productScope has only one key defined, it is the productID	{
			"key": "productID",
			"value":["Foxtel_Hunger_Games_PROD1"]
			,
			}

4.1.1.3 criteria

Property	Type	Description	Sample	Notes
key	string	attribute in which we apply the criteria	Actor	For the moment, no list of available keys is provided. In this case, there is no validation if this key exists or not.
operator	string	Operator we apply between the key and the value, if no value consider as equal operator	-	
value	List<string>	value of the attribute	["Jennifer"]	In this case the list has only one element

4.1.1.4 accountScope

Property	Type	Description	Sample
logicalOperator	string	logical operator used between each criterias logicalOperator is not mandatory. If not provided, it will be considered as a AND	"logicalOperator": "$or"

criterias	criteria
	(maxOccurs="unbounded
	")
		Composed of key, value list and operator	{
			"key":
			"accessPoint",
			"value":["France"],
			}

4.1.1.5 subscriberProfile

Property	Type	Description	Sample
logicalOperator	string	logical operator used between each criterias	"logicalOperator": "$or"

criterias	criteria
	(maxOccurs="unbounded")
		Composed of key, value list and operator	{
			"key": "spec1",
			"value":["val1"],
			}

4.1.1.6 Operator list

you will find below the list of operators that can be applied between the key and value(s). Note that for the moment, there is no list of available keys provided. In this case, there is no validation of keys provided in the system.

Operator	Description	OMS Syntax in JSON	Meaning	Applicability

-	Equals	{
		"key" : "price",
		"value" : ["1.99"]
		}
			price = 1.99	numbers, string, date

$gt	Matches values that are greater than the value specified in the query.	{
		"key" : "price",
		"value" : ["1.99"],
		"operator" : "$gt"
		}
			price > 1.99	numbers, string, date

$gte	Matches values that are equal to or greater than the value specified in the query.	{
		"key" : "price",
		"value" : [1.99],
		"operator" : "$gte"
		}
			price >= 1.99	numbers, string, date

$in	Matches any of the values that exist in an array specified in the query	{ "key" : "genre",	when genre is drama OR action OR sport	string, data

		"value" : ["drama", "action", "sport"], "operator" : "$in" }		

`$lt`	Matches values that are less than the value specified in the query.	{
		"key" : "price",
		"value" : ["1.99"],
		"operator" : "$lt"
		}
			price < 1.99	numbers, string, date

`$lte`	Matches values that are less than or equal to the value specified in the query.	{
		"key" : "price",
		"value" : ["1.99"],
		"operator" : "$lte"
		}
			price =< 1.99	numbers, string, date

`$ne`	Matches all values that are not equal to the value specified in the query	{
		"key" : "genre",
		"value" : ["drama"],
		"operator" : "$ne"
		}
			all content with genre <> drama	numbers, string, date

`$nin`	Matches values that **do not** exist in an array specified to the query.	{
		"key" : "genre",
		"value" : ["drama", "action", "sport"],
		"operator" : "$nin"
		}
			when genre is not drama OR action OR sport	numbers, string, date

4.2 Promotion examples

Promotion with a content scope

```
{
    "promotions": [{
        "title": "title_015",
        "description": "description_015",
        "serviceProvider": "String",
        "startAvailability": "1383048000000",
        "endAvailability": "1385726400000",
        "enabled": "true",
        "discountType": "percentage",
        "discountValue": 20,
        "billingPeriods": 3,
        "scope": {
            "contentScope": {
                "logicalOperator": "$or",
                "criterias": [
                {
                    "criteria": [
                    {
                        "key": "startAvailability",
                        "value": ["2013-11-25"],
                        "operator": "$gte"
                    },
                    {
                        "key": "endAvailability",
                        "value": ["2013-11-30"],
                        "operator": "$lte"
                    }
                    ]
                },
                {
                    "criteria": [
                    {
                        "key": "startAvailability",
                        "value": ["2013-12-25"],
                        "operator": "$gte"
                    },
                    {
                        "key": "endAvailability",
                        "value": ["2013-12-30"],
                        "operator": "$lte"
                    }
                    ]
                }
                ],
            "productType": "single"
            }
        }
    }
    ]
}
```

Promotion with a productScope

```
{ "promotions": [{
    "title": "title_021",
    "description": "description_021",
    "serviceProvider": "SP1",
    "startAvailability": "1383048000000",
    "endAvailability": "1385726400000",
    "enabled": "true",
    "discountType": "percentage",
    "discountValue": 20, "billingPeriods": 3,
    "scope": {
        "productScope": {
            "logicalOperator": "$or",
            "criterias": [
                {
                    "criteria": [
                        {
                        "key": "productID",
                        "value": ["Foxtel_Inception_PROD1"]
                        }
                    ]
                },
                {
                    "criteria": [
                        {
                        "key": "productID",
                        "value": ["Foxtel_Hunger_Games_PROD1",
"Foxtel_Hunger_Games2_PROD"],
                        "operator": "$in"
                                }
                    ]
                }
                ]
            }
        }
    }
    ]
}
```

4.3 Operations available

Based on the WADL provided we have the following operations available for the notification:

- POST for create
- GET for retrieve
- PUT for update
- DELETE for delete

Functionality	URL	Payload	Response
POST	…/promotions/<SP>	promotions	OMSResponse
GET	…/promotions/<SP>/<ID>	-	Promotions
GET	…/promotions/<SP>/<query_string>	-	Promotions
PUT	…/promotions/<SP>/<ID>	Promotions	OMSResponse
DELETE	…/promotions/<SP>/<ID>	-	OMSResponse
DELETE	…/promotions/<SP>/<query_string>	-	OMSResponse

4.4 Descriptions of each operation:

POST	**.../promotions/<serviceProvider>/**

GOAL: Create a new promotion object and returns an OMSResponse object (see table blelow).

Path Param:

serviceprovider: The service provider name. If the Service provider value is not existing in the DB, then an error must be thrown.

Query Param: none.

Payload: will carry a promotions object (without service provider which is already given in the URL, and without ID which is generated) that contain one or several promotion objects. If an ID or a service provider value is provided within one of the promotions, an error should be returned within the response. If a "serviceProvider" value is provided within the payload it must match the one in the URL otherwise an error must be thrown.

Response Entity:

OMSResponse:

statusCode

statusMessage

ids = The ids of the created promotions.

total_records = The number of created promotions

GET	**.../promotions/{serviceProvider}/{id}**

GOAL: Retrieve ONE specific promotion given by the service provider and the promotion Id

Path Param:

serviceProvider: The service provider name. If the Service provider value is not existing in the DB, then an error must be thrown.

id: The promotion Id (unique id associated to the promotion)

Query Param:

limit: The maximum number of objects that should be returned by this query

offset: The page offset starting from zero

Payload: none.

Response Entity: A promotions object that wraps a Promotion Item. The promotions object contains even the following properties:

total_records= total number of promotion retrieved in this case = 1

offset = page offset starting from zero

version = version of the data (not yet implemented in these sprints)

promotions = The retrieved promotion object

PUT	**.../promotions/{serviceProvider}/{id}**

GOAL: Update a given promotion by the service provider and id.

Path Param:

serviceProvider: The service provider name. If the Service provider value is not existing in the DB, then an error must be thrown.

id: The promotion Id (unique id associated to the promotion)

Query Param: none.

Payload: A Promotions object with only one promotion inside defining the new values to be set to the retrieved promotion. If a "serviceProvider" value is provided within the payload it must match the one in the URL otherwise an error must be thrown.

Response Entity:

OMSResponse:

statusCode

statusMessage

ids = empty.

total_records = The number of updated promotions, 1 in this case.

DELETE	**.../promotions/{serviceProvider}/{id}**

GOAL: Delete a single promotion given by the service provider and the id.

Path Param:

serviceProvider: The service provider name. If the Service provider value is not existing in the DB, then an error must be thrown.

id: The promotion Id (unique id associated to the promotion)

Query Param: none.

Payload: none.

Response Entity:

OMSResponse:

statusCode

statusMessage

ids = empty.

total_records = The number of deleted promotions, 1 in this case.

GET	**.../promotions/{serviceProvider}?<query_string>**

GOAL: Retrieve all promotions managed and stored by the OMS with filter expressed by the provided parameters.

Path Param:

serviceProvider: The service provider name. If the Service provider value is not existing in the DB, then an error must be thrown.

Query Param (all joined by an &):

limit: The maximum number of objects that should be returned by this query

offset: The page offset starting from zero

id: The ID of the searched promotion

title: The title of the searched promotion

description: The description of the searched promotion (should be strictly case sensitive equal)

startAvailability: The start availability of the search promotion

endAvailability: The end availability of the searched promotion

enabled: Whether or not the searched promotion is enabled (activated).

discountType: The discount type of the searched promotion (*percentage* / *finalPrice*)

discountValue: The discount value of the searched promotion (associated to the discount type)

billingPeriods: The number of billing periods of the searched promotion.

priority: The priority of the searched promotion

Payload: none.

Response Entity: A promotions object that wrap a list of promotion items. The promotions object contains even the following properties:

total_records= total number of promotion retrieved

offset = page offset starting from zero

version = version of the data (not yet implemented in these sprints)

promotions = The retrieved promotion objects

PUT	**.../promotions/{serviceProvider}?<query_string>**

GOAL: Update the properties of all the promotion objects stored in the db that corresponds to the given search criteria.

Path Param:

serviceProvider: The service provider name. If the Service provider value is not existing in the DB, then an error must be thrown.

Query Param (all joined by an &):

id: The ID of the searched promotion

title: The title of the searched promotion

description: The description of the searched promotion (should be strictly case sensitive equal)

startAvailability: The start availability of the search promotion

endAvailability: The end availability of the searched promotion

enabled: Whether or not the searched promotion is enabled (activated).

discountType: The discount type of the searched promotion (*percentage* / *finalPrice*)

discountValue: The discount value of the searched promotion (associated to the discount type)

billingPeriods: The number of billing periods of the searched promotion.

priority: The priority of the searched promotion.

Payload: A Promotions object with only one promotion inside defining the new values to be set to the retrieved promotions. If a "serviceProvider" value is provided within the payload it must match the one in the URL otherwise an error must be thrown.

Response Entity:

OMSResponse:

statusCode

statusMessage

ids = empty

total_records = The number of updated promotions

DELETE	**.../promotions/{serviceProvider}?<query_string>**

GOAL: Delete all the promotion objects stored in the db that corresponds to the given search criteria.

Path Param:

serviceProvider: The service provider name. If the Service provider value is not existing in the DB, then an error must be thrown.

Query Param (all joined by an &):

id: The ID of the searched promotion

title: The title of the searched promotion

description: The description of the searched promotion (should be strictly case sensitive equal)

startAvailability: The start availability of the search promotion

endAvailability: The end availability of the searched promotion

enabled: Whether or not the searched promotion is enabled (activated).

discountType: The discount type of the searched promotion (*percentage* / *finalPrice*)

discountValue: The discount value of the searched promotion (associated to the discount type)

billingPeriods: The number of billing periods of the searched promotion.

priority: The priority of the searched promotion

Payload: none.

Response Entity: OMSResponse object with the following properties:

OMSResponse:

statusCode

statusMessage

ids = empty.

total_records = The number of deleted promotions.

5. Notification

The scope of this interface is to notify an OMS consumer.
The base URL definition for managing the Notification is defined with the endpoint and the WADL. And it is:

```
http://<host>:port/cxf/oms/omm/notification/<version>/notifications
```

- Host is the OSGi host name
- Port of the host, by default is 8181
- Version of the interface. It is 1 for the first release

5.1 Operations available

Based on the WADL provided we have the following operations available for the promotions:

- GET for retrieve

Functionality	URL	Payload	Response
GET	…/notifications/<SP>?<query_string>	-	notifications
GET	…/notifications/<SP>/currentSequenceNumer	-	number
GET	…/notifications/<SP>/<sequenceNumber>	-	notifications

5.2 Descriptions of each operation:

GET	**.../notifications/{serviceProvider}?<query-string>**

GOAL: Retrieve all the notifications for the given service provider other provided criteria

Path Param:

serviceProvider: The service provider name. If the Service provider value is not existing in the DB, then an error must be thrown.

Query Param:

since: The sequence number value from which the notifications are searched, so as all the notifications having a sequence number strictly greater than the given one will be returned.

limit: The maximum number of objects that should be returned by this query

offset: The page offset starting from zero

Payload: none.

Response Entity: A notifications object that wraps notification Items. The notifications object contains even the following properties:

total_records= total number of promotion retrieved

offset = page offset starting from zero

seqenceNumber: The latest notification sequence number of the given service provider

notifications = The retrieved notification objects

GET **.../notifications/{serviceProvider}/currentSequenceNumber**

GOAL: retrieve the sequence number of the latest created notification

Path Param:

serviceProvider: The service provider name. If the Service provider value is not existing in the DB, then an error must be thrown.

Query Param: none.

Payload: none.

Response Body: The sequence number of the latest created notification.

GET **.../notifications/{serviceProvider}/{sequenceNumber}**

GOAL: Retrieve the notification for the given service provider and provided sequence number.

Path Param:

serviceProvider: The service provider name. If the Service provider value is not existing in the DB, then an error must be thrown.

sequenceNumber: The searched notification sequence number.

Query Param: none.

Payload: none.

Response Entity: A notifications object that wraps notification Items. The notifications object contains even the following properties:

total_records= total number of promotion retrieved in this case = 1

offset = page offset starting from zero (always 0 in this case, as pagination feature is useless here)

seqenceNumber: The latest notification sequence number of the given service provider *notifications* = The retrieved notification object

6. Service Provider

The base URL definition for managing the Notification is defined with the endpoint and the WADL. And it is:

```
http://<host>:<port>/cxf/oms/omm/sp/<version>/
```

- Host is the OSGi host name
- Port of the host, by default is 8181
- Version of the interface. It is 1 for the first release

6.1 Operations available

Based on the WADL provided we have the following operations available for the service provider:

- GET for retrieve
- POST for create

Functionality	URL	Payload	Response
GET	.../<SP>?<query_string>	-	serviceProviders
POST	.../<SP>	serviceProviders	OMSResponse

6.2 Descriptions of each operation:

GET	**.../serviceproviders?<query-string>**

GOAL: Retrieve ONE specific service provider given by the name. Or retrieve a list of service provider if the name parameter is not given.

Query Param:

name: The name of the service provider

Payload: none.

Response Entity: A serviceProviders object that wraps a list service provider Item. The serviceProviders object contains even the following properties:

total_records= total number of promotion retrieved in this case = 1

offset = page offset starting from zero

version = version of the data (not yet implemented in these sprints)

serviceProviders = The retrieved service provider list

POST	**.../serviceproviders**
GOAL: Create one or several service provider objects **Query Param:** None **Payload:** A serviceProviders object that wraps several serviceProvider objects. **Response Entity:** *OMSResponse*: *statusCode* *statusMessage* *ids* = empty *total_records* = The number of created service providers	

6.3 Service Provider example

The payload of serviceProviders is the following:

```
{"serviceProviders":
[{"name":"SP1",
 "description":"description_002"
}]
}
```

—— END OF DOCUMENT ——