Cloud Application Framework

Version 1.0STD12

User Guide

Filename	CAF_USE_UserGuide_1.0STD12_pub.pdf
Document Nr	CAF-USE
Date	2019-05-22
Author(s)	KUDELSKI SA
Information domain	
Data Owner	Nagra Media UK Limited

KUDELSKI SA
22-24, Route de Geneve, Case Postale 134, 1033 Cheseaux , Switzerland
tel: +41 21 732 01 01, +41 21 732 01 00
http://www.nagra.com/

Tracking data

Contributors

Event	Actor	Date	Notes
reviewer	Matt Lucas	2018-02-22	
approver	Matt Lucas	2018-02-22	

Reviewers

Reviewer	Date	Notes

Approvers

Approver	Date	Notes
Matt Lucas	2018-02-22	

Contents

1 Overview. 6
2 Release Notes. 7
2.1 Release 1.0.12. . . . . *New* 7
2.2 Release 1.0.8. . . . . *New* 7
2.3 Release 1.0.7. 8
2.4 Release 1.0.6. 10
2.5 Release 1.0.5. 11
2.6 Release 1.0.3. 12
2.7 Release 1.0.4. 13
3 Administration. 15
3.1 Installation. 15
3.2 Management. 16
3.3 Puppet. 17
3.4 CAF as an MDS Decorator. . . . . *New* 18
3.5 Install and Configure Redis Cache. . . . . *New* 20
4 Configuration. 23
4.1 Format. 23
4.2 Plugins. 26
5 SDK API. 29
5.1 Redis. 29
5.2 MDS. 31
5.3 SDP. 33
5.4 Third Party Node Modules. 35
6 Bundled Plugins. 38
6.1 Ping. 38
6.2 Service Subscription. 38
6.3 Node Count. 41
6.4 Image Transformer. 44

6.5 CMS4-to-NXgen. 46
6.6 VoD Series-Product Aggregation. 54
6.7 Caching Channel Facade. . . . . *New* 56
7 Best Practices. 62
8 Implementing a CAF Module. 67
9 Foxtel. 70
9.1 Admin APIs RAML. 70
9.2 Circuit Breaker. 71
9.3 Client APIs RAML. 71
9.4 Current State of Release 1.0.8. . . . . *New* 74
9.5 HTTP Router. 74
9.6 Puppet deployment. 74
9.7 Purchase APIs. 75
9.7.1 DELETE - /caf/purchase/admin/purchases/cache. 75
9.7.2 DELETE - /caf/purchase/admin/purchases/cache/<accountNumber>. 75
9.7.3 GET - /caf/purchase/purchases. 75
9.7.4 GET - /caf/purchase/purchases/contentDrm/<contentDrmId>. 76
9.7.5 POST - /caf/purchase/purchases/product/<productId>. 77

1 Overview

Client UI Developers are often frustrated by what they see as limitations in the Media Live Service Platform services. It cannot always fulfill each of the required use cases of each client application in a single API request, as catering for the specific needs of each individual project's client UI application would result in an expensive, slow and bloated system. Meanwhile, we also do not want to branch our products and create customer specific code. We need our product to cater for the common use cases across all projects in way that is scalable and efficient.

This product is an implementation of a server side service that can act as a facade for other media live service platform services. To enable client UI application developers to easily add to this service the framework is implemented using a server side JavaScript framework utilising NodeJS.

Using the NodeJS framework will allow client developers to use the same skills, and in some cases, the same code as they use in client application development.

2 Release Notes

2.1 Release 1.0.12 *New*

Here are some important notes related to this release

Highlights of Cloud Application Framework 1.0.12

- Addition of Caching Channel Facade

Puppet Config Changes

Param	Change Description	Old Default	New Default	ActionRequired
redis_sentinels	A comma-list of host:port redis sentinel members.			Consider

2.2 Release 1.0.8 *New*

Here are some important notes related to this release

Highlights of Cloud Application Framework 1.0.8

- Fixing of missing field from MDS calls

Upgrade Notes

- None.

Puppet Config Changes

Param	Change Description	Old Default	New Default	ActionRequired
None				

MLM Config Changes

Param	Change Description	Old Default	New Default
None			

End of support announcements

None

Issues List

Fixed Issues

Issue No	Status	Summary

2.3 Release 1.0.7

Below are some important notes related to this release

Highlights of Cloud Application Framework 1.0.7

- Introduction of new admin APIs to delete the https://engwiki/pages/viewpage.action?pageId=129864782 or https://engwiki/pages/viewpage.action?pageId=129864786.
- Introduction of https://engwiki/display/CAF/Circuit+Breaker.
- Introduction of integration tests for the Federated layer.
- Introduction of new puppet parameter caf_foxtel_admin_ip_list which can be used to add permitted list of IPs to access the admin APIs

Upgrade Notes

- None.

Puppet Config Changes

Param	Change Description	Old Default	New Default	ActionRequired
foxtelCircuitBreaker Enabled	Whether the circuit breaker is enabled	N/A	false	
foxtelCircuitBreaker Timeout	Maximum time to wait before rejecting request	N/A	3000	
foxtelCircuitBreaker ErrorThreshold Percentage	Minimum percentage of failed requests to trip circuit	N/A	50	
foxtelCircuitBreaker ResetTimeout	Time taken for circuit breaker to change to half-open from open	N/A	30000	
caf_foxtel_admin_ ip_list	List of APIs allowed to access Admin APIs	N/A	[]	

MLM Config Changes

Param	Change Description	Old Default	New Default
None			

End of support announcements

None

Issues List

Fixed Issues

Issue No	Status	Summary

2.4 Release 1.0.6

Below are some important notes related to this release

Highlights of Cloud Application Framework 1.0.6

- Changes to the API calls to SDP and makes use of the SDP-API adaptor.
- Much improved unit tests and inclusion of integration-tests utilizing a docker environment and mocked external services.
- Nginx at medialive layer now also makes calls to SDP to retrieve the accountUid and incorporates this into the token created.

Upgrade Notes

- None.

Puppet Config Changes

Param	Change Description	Old Default	New Default	ActionRequired
None				

MLM Config Changes

Param	Change Description	Old Default	New Default
None			

End of support announcements

None

Issues List

Fixed Issues

Issue No	Status	Summary

2.5 Release 1.0.5

Below are some important notes related to this release

Highlights of Cloud Application Framework 1.0.5

- Foxtel Federated Head End changes - complete workflow between the cache layer and the ML layer
- Foxtel Media Live service is now available. Tested only with Mocks (SDP and Purchase Server)
- Nginx support for whitelists (Fed HE) and token generation (Media Live)

Upgrade Notes

- None.

Puppet Config Changes

Param	Change Description	Old Default	New Default	ActionRequired
None				

MLM Config Changes

Param	Change Description	Old Default	New Default
None			

End of support announcements

None

Issues List

Fixed Issues

Issue No	Status	Summary

2.6 Release 1.0.3

Below are some important notes related to this release

Highlights of Cloud Application Framework 1.0.3

- Update to CMS4 NG Adapter to conform to new CPM specification.

Upgrade Notes

- None.

Puppet Config Changes

Param	Change Description	Old Default	New Default	ActionRequired
None				

MLM Config Changes

Param	Change Description	Old Default	New Default
None			

End of support announcements

None

Issues List

Fixed Issues

Issue No	Status	Summary

2.7 Release 1.0.4

Below are some important notes related to this release

Highlights of Cloud Application Framework 1.0.4

- Foxtel pre-release contains new purchases API

Upgrade Notes

- None.

Puppet Config Changes

Param	Change Description	Old Default	New Default	ActionRequired
None				

MLM Config Changes

Param	Change Description	Old Default	New Default
None			

End of support announcements

None

Issues List

Fixed Issues

Issue No	Status	Summary

3 Administration

3.1 Installation

Installation of the cloud application service is as simple as installing a few related rpms.

Tip

It is highly recommended that you use puppet as the primary installation method.

Only follow these manual steps if your deployment does not support puppet, or for debugging / testing purposes.

Installing Redis

Install Redis

The first step is to install the standard nagra Redis package:

```
rpm -i redis-3.0.1.x86_64.rpm
```

Tip

If you already have Redis installed for other modules, and it is appropriate, you can skip this step

Install CAF Redis Config

As Redis is a lightweight, (essentially) single threaded, process each module can provide configuration to create a customized Redis instance specifically for that module.

You can find the CAF configuration within the "caf-redis" package, so please go ahead and install this:

```
rpm -i caf-redis-3.2.0.noarch.rpm
```

This simply drops a Redis configuration file into the following location:

```
/opt/redis/etc/redis_cafcache.conf
```

To start this specific version of Redis, just run:

```
service redis start cafcache
```

Now repeat for every node in your cluster.

Installing node.js

Simply install the node.js base rpm:

```
rpm -i nagra-nodejs-base-0.12-STD0.x86_64.rpm
```

Installing Cloud Application Framework

Again, simply install the cloud-application-framework rpm, and it's related Redis rpm, which contains Redis specific configuration:

```
rpm -i cloud-application-framework-1.0-STD0.noarch.rpm
rpm -i cloud-application-framework-REDIS-1.0-STD0.noarch.rpm
```

Once the above have been successfully installed, please restart the following services to apply the associated config.

```
service redis restart
service nginx restart
```

Ensuring Configuration

Before starting the cloud application framework you should make sure that the following key configuration sections have been checked and corrected as appropriate.

- MDS.browseEndpoint
- SDP.URL
- Redis.host
- Redis.port
- Registry.*

Caution! Please see the configuration section for more specific details on these sections

3.2 Management

Start

Simply starts the application:

```
service caf start
```

when doing this you should see an output such as the following.

```
Starting cloud-application-framework
[PM2] Spawning PM2 daemon
[PM2] PM2 Successfully daemonized
```

```
[PM2] Starting /opt/cloud-application-framework/caf.js in cluster_mode (0 instance)
[PM2] Done.
┌──────────────────────────┬────┬─────────┬──────┬────────┬─────────┬────────┬─────────┬──────────┐
│ App name                 │ id │ mode    │ pid  │ status │ restart │ uptime │ memory  │ watching │
├──────────────────────────┼────┼─────────┼──────┼────────┼─────────┼────────┼─────────┼──────────┤
│ cloud-application-framework │ 0  │ cluster │ 25953 │ online │ 0       │ 1s     │ 40.594 MB │ disabled │
│ cloud-application-framework │ 1  │ cluster │ 25962 │ online │ 0       │ 1s     │ 40.508 MB │ disabled │
│ cloud-application-framework │ 2  │ cluster │ 25971 │ online │ 0       │ 1s     │ 40.930 MB │ disabled │
│ cloud-application-framework │ 3  │ cluster │ 25980 │ online │ 0       │ 0s     │ 40.805 MB │ disabled │
│ pm2-http-interface       │ 4  │ fork    │ 26049 │ online │ 0       │ 0s     │ 6.262 MB  │ disabled │
└──────────────────────────┴────┴─────────┴──────┴────────┴─────────┴────────┴─────────┴──────────┘
```

The table above lists the processes that have been started, an instance of the "cloud-application-framework" for each physical core on the machine (to scale across the machine), and a single pm2-http-interface process which allows health-checks, and the like, for the processes.

Stop

Simply stops the application and the monitoring interface:

```
service caf stop
```

Status

Outputs the same status table as shown above:

```
service caf status
```

Reload

Does a zero-downtime reload of the processes, for example, after a configuration change.

```
service caf reload
```

3.3 Puppet

Setup

To allow cloud-application-framework to be deployed through puppet and foreman, there are a few pre-requisites :

- Install the cloud-application-framework-MLDS module onto the foreman server.

  Instructions on how to install MLDS modules are available on the MLDS_USE_InstallationandUserGuide, which is part of the MLS-Server release.
- nagra-nodejs-base rpm should have been pushed into the yum repo.
- cloud-application-framework rpm should have been pushed into the yum repo.
- Make sure that all of your servers have their yum repositories configured to point at one containing all relevant Nagra modules.

Deploying cloud-application-framework Using Foreman

Once installed, you should see the following class listed :

1. You can select this class for installation.

2. Then, the smart class parameters must be configured in Foreman in order for the cloud-application-framework to install and run successfully.

cloud-application-framework

1. Installs the cloud-application-framework software, dependencies and supporting files onto the server, e.g. nagra-nodejs-base

2. This module ensures that cloud-application-framework isrunning at all times

Configuration

All configuration items for cloud-application-framework are exposed under the parameters tab in the foreman GUI and have extra information in the 'Additional info' popup. For example:

3.4 CAF as an MDS Decorator *New*

A number of CAF modules are designed to augment existing MDS API's in ways that cannot be natively supported. It is usually desirable for this functionality to be added to the the exposed MDS API without having to make any change on the client side. In other words, the client shouldn't be aware that it is talking to anything other than MDS.

It is possible to achieve this decoration easily through nginx configuration, and the following guide will take you through the steps to achieve this. For the example let's use the **cu-programmes-enhanced** plugin.

Setting up CAF Configuration

To transparently wrap MDS, there are a couple of tweaks we need apply to the CAF configuration to ensure it is mapped correctly on both the front and back-end. The following instructions can be applied to the CAF configuration either directly or via puppet, as your deployment dictates.

Remove Base URL

To imitate an MDS endpoint, without complex nginx remapping, we need to remove the CAF specific base URL.

```
"baseUrl": ""
```

Set MDS.browseEndpoint

Any delegation to MDS needs to go directly, and not via nginx, to avoid a cycle in the call chain.

```
"browseEndpoint": "http://127.0.0.1:12123/metadata/delivery/EUSKALTEL"
```

Enable Appropriate Plugin

The decorating plugin should be mapped onto the exact MDS endpoint that it simulates.

```
"/metadata/delivery/EUSKALTEL/btv/programmes": "./lib/cu-programmes-enhanced"
```

Configuring Nginx

Note

Please enter any custom nginx configuration in the provided **`local.conf`** files. These files are protected during future updates or configuration of nginx.

Configuring CAF Upstream

To control the service to which our API call will be sent we need to create an upstream configuration. Please enter this information into the **`/opt/nginx/conf/http/local.conf`** file.

(Optional) If the feature that the plugin provides isn't strictly mandatory then it is a good idea to add the original MDS as a backup server. This will allow for graceful degradation in the case that CAF is not able to provide a service by sending the request to vanilla MDS instead.

```
upstream caf_mds_override {
    server 127.0.0.1:3000;
    server 127.0.0.1:12123 backup;
    keepalive 50;
}
```

Overriding Specific Endpoint(s)

To redirect a specific endpoint (and not all MDS endpoints) to CAF, we simply need to provide a location block that is more specific than those MDS itself provides.

Try adding this to the file **`/opt/nginx/conf/server_80/local.conf`**

```
location /metadata/delivery/EUSKALTEL/btv/programmes {
    proxy_pass http://caf_mds_override;
}
```

Now restart / reload nginx, and your service should be available.

References

- Decorator pattern - https://en.wikipedia.org/wiki/Decorator_pattern

3.5 Install and Configure Redis Cache *New*

Install Redis

The first step is to install the standard nagra Redis package:

```
rpm -i redis-3.0.1.x86_64.rpm
```

Tip If you already have Redis installed for other modules, and it is appropriate, you can skip this step

Install SRM Redis Config

As Redis is a lightweight, (essentially) single threaded, process each module can provide configuration to create a customized Redis instance specifically for that module.

You can find the CAF configuration within the "caf-redis" package, so please go ahead and install this:

```
rpm -i cloud-application-framework-REDIS-x.y.z.noarch.rpm
```

This simply drops a Redis configuration file into the following location:

```
/opt/redis/etc/redis_cafcache.conf
```

To start this specific version of Redis, just run:

```
service redis start cafcache
```

Now repeat for every node in your cluster.

Tip It is recommended that you have at least 3 Redis nodes in your cluster

Setup Replication

Once all nodes are up and running, we need to connect them as [master 1-N slaves], to allow for replication.

First, identify the node you want as your immediate master.

Next, iterate through all slave nodes (everyone else) and run the following commands:

```
/opt/redis/bin/redis-cli -p 5570
slaveof "172.16.8.83" 5570
```

Caution! Obviously replace the host and port with those relevant to your deployment

Once complete, check the master to see that the replication has been successful:

```
/opt/redis/bin/redis-cli -p 5570
info
```

and you should see output similar to the following, describing the other nodes we have just visited:

```
# Replication
role:master
connected_slaves:2
slave0:ip=172.16.8.84,port=5570,state=online,offset=197,lag=1
slave1:ip=172.16.8.85,port=5570,state=online,offset=197,lag=0
```

Configure Sentinel

Redis Sentinel provides high availability for Redis. In practical terms this means that using Sentinel you can create a Redis deployment that resists without human intervention to certain kind of failures.

The caf-redis package comes with a snippet that allows you to configure Redis Sentinel to look after our SRM nodes. It should look something like this:

```
sentinel monitor srmcache <IP> 5570 2
sentinel down-after-milliseconds cafcache 2000
sentinel failover-timeout cafcache 18000
sentinel parallel-syncs cafcache 8
sentinel config-epoch cafcache 216
```

Simply replace <IP> with that of your cafcache master, and restart the sentinel

```
service redis_sentinel stop
service redis_sentinel start
```

Do this on every node.

Now to verify our setup.

```
sentinel masters
```

Checks that our master node appears in the list, e.g.

```
1) "name"
2) "cafcache"
3) "ip"
4) "172.16.8.83"
```

Now the slaves:

```
sentinel slaves cafcache
```

should show all other nodes:

```
1)  1) "name"
    2) "172.16.8.85:5560"
2)  1) "name"
    2) "172.16.8.84:5560"
```

And finally, the sentinels

```
sentinel sentinels cafcache
```

should show all sentinel nodes (other than the current one):

```
1)  1) "name"
    2) "172.16.8.84:26379"
2)  1) "name"
    2) "172.16.8.85:26379"
```

4 Configuration

4.1 Format

Server

Field	Type	Description
port	int	The port to which this service should bind
baseUrl	string	The base URL to which all other urls are mapped, e.g. /<base>/...

MDS

Field	Type	Description
browseEndpoint	string	The MDS base endpoint. **Caution!** In a clustered environment should be mapped through some kind of load balancer for redundancy of upstream services
vod.nodes.cache PrimerQuery	string	An open MDS query that will prime the nodes cache for certain calls.
vod.nodes.request NodeLimit	int	Specifies maximum number of node requests that may be run in parallel.
vod.nodes.requestSub NodeLimit	int	Specifies maximum number of sub-node requests that may be run in parallel.
vod.nodes.defaultMax Age	int	The default max-age to add in the Cache-Control head on response (if none returned from upstream).
poolSize	int	The maximum number of connections held open to MDS.

SDP

Field	Type	Description
URL	string	The SDP base endpoint. **Caution!** In a clustered environment should be mapped through some kind of load balancer for redundancy of upstream services
PATH	string	Relative base path of SDP services, e.g. /qsp/gateway/http/js/
ADAPTOR_PATH	string	Relative base path of API Adaptor services, e.g. /adaptor/hue-gateway/gateway/http/js/

Redis

Field	Type	Description
host	string	The ip of the redis service
port	int	The port of the redis service
socketKeepalive	int	Amount of time to keep alive an idle socket
connectionPoolSize	int	Max size of redis connection pool

Registry

Field	Type	Description
<url>*	url => filepath	A route mapping of url endpoint to a file path for a plugged-in module. **Caution!** All endpoints specified here are relative, and will be prefixed by the baseUrl as configured within the Server section. For example /ping will become /caf/ping by default.

Field	Type	Description
		Warning! Please only register any modules you want to be available to your service, and remove any others. This will save on computing resource and make debugging/logging simpler.

ImageTransformer

Field	Type	Description
tmpFolder	string	Where to store source images locally
cacheFolder	string	Where to store modified images locally
tmpFolderTTL	int	How long to keep source images for
cacheFolderTTL	int	How long to keep modified images for
checkPeriod	int	How often to check for expired images
quality	int	The quality required in terms of %, so 0-100.
interlace	string	The interlacing scheme None/Line/Plane/Partition

CMS4Adapter

Field	Type	Description
billingModel.period	string	Default period for the billing model in adapted result.
periodRounding	int	The nearest seconds to which startAvailabilityOffset and endAvailabilityOffset will be rounded for upstream cachability purposes.
locale	string	The default locale by which CAF should query MDS for the adapter.

Field	Type	Description
selfPort	string	The port for self references.
cacheMaxAge	int	The value to include in the Cache-Control:max-age header.

4.2 Plugins

Plugin Configuration

The plugin framework for cloud-application-framework is simple and easy to use.

The registry configuration is just a simple route, from the URL endpoint you wish to plug in to, to the module you wish to plug in to it.

Field	Type	Description
<url>*	url => filepath	A route mapping of url endpoint to a file path for a plugged-in module. **Caution!** All endpoints specified here are relative, and will be prefixed by the baseUrl as configured within the Server section. For example /ping will become /caf/ping by default. **Warning!** Please only register any modules you want to be available to your service, and remove any others. This will save on computing resource and make debugging/logging simpler.

Simple "Hello" Plugin Deployment Example

Overview

In this walk-through, we will make a simple plugin and demonstrate one kind of deployment strategy and configuration.

Caution! Other deployment methods are available, this is just one suggested path.

The plugin we will create is a simple hello world response - or if you provide a parameter name=..., it will respond by saying hello to that name.

Create a Simple plugin

First thing, is to create a "hello" directory and initialize the npm project:

```
mkdir hello; cd hello
npm init
```

Once the wizard has completed, we will have our generated **`package.json`**, and we just want to create our plugin in an **`index.js`** file. Fill that file with the following express route:

```
var express = require('express');
var router = express.Router();
router.get('/', function (req, res) {
  var name = req.query.name || "world";
  res.send("hello " + name + "!");
});
module.exports = router;
```

Last thing is just to package this up into a tarball with:

```
npm pack
```

Now you should have a package appear **`hello-1.0.0.tgz.`**

Install the Plugin

To install the plugin on each cloud-application-framework instance, simply copy the file onto the box, and run a command similar to the following:

```
export PATH=$PATH:/opt/nodejs-base-4.2/bin/
cd /opt/cloud-application-framework/
su appaccelerator -c "npm install --prefix /opt/cloud-application-framework/plugins/
 hello-1.0.0.tgz"
```

The plugin should be expanded in the directory **`/opt/cloud-application-framework/plugins.`**

Add the Plugin to the Registry

Once installed, simply edit the Registry configuration, and add a route for your new module:

```
"Registry": {
   "/img": "./lib/image-transformer",
   "/btv": "./lib/btv",
   "/ping": "./lib/ping",
   "/hello": "./plugins/node_modules/hello"
```

```
}
```

Reload Cloud Application Framework

Now reload caf to apply the changes:

```
service caf reload
```

You should now be able to query the plugin through the API:

```
hello world!
```

```
 hello dave!
```

5 SDK API

The following API descriptions are of internal utilities, that may aid plugin development, rather that HTTP interfaces.

5.1 Redis

The Redis library contains some useful caching functions, and manages all the general connection pooling and other administrative functions so you don't have to.

Importing

Simply import as follows:

```
var RedisNamespace = require('../redis').RedisNamespace
```

Create Redis Namespace

Firstly, you will need to create a custom Redis namespace so that your cache isn't conflicting with other caches in the cloud-application-framework.

Caution! Unless explicitly doing so, please make sure that your namespace name is sufficiently unique to not clash with others - e.g. don't just use "cache" for example.

```
var responseCache = new RedisNamespace("noderes");
```

Cache Operations

There are a few simple operations you can make to a cache, described as follows:

get (key, fn)

Gets the corresponding value to the passed in key, asynchronously.

Argument	Type	Description
key	String	The key for the lookup, excluding any name-spacing
fn	Callback Function	A callback function that is called on lookup completion of form: `function(error, result)`

set (key, value)

Asynchronously sets the given key to the given value.

Argument	Type	Description
key	String	The key for the lookup, excluding any name-spacing
value	String	The value to map to the given key

setWithExpiry (key, value, ttl)

Asynchronously sets the given key to the given value, with a predefined expiry.

Argument	Type	Description
key	String	The key for the lookup, excluding any name-spacing.
value	String	The value to map to the given key.
ttl	Int	The number of seconds after which the value will be automatically removed from the cache.

getOrElse (key, fn, els)

A convenience method over the standard get. Avoids null checking on caller side, by specifying an alternative instead.

Argument	Type	Description
key	String	The key for the lookup, excluding any name-spacing
fn	Callback Function	A callback function that is called on lookup completion of form: `function(error, result)`
els	Callback Function	If no value is found, uses this callback instead. `function(error, result)`

expire (key, ttl)

Sets an expiry on the given key with the given ttl.

Argument	Type	Description
key	String	The key for the lookup, excluding any name-spacing.
ttl	Int	The number of seconds after which the value will be automatically removed from the cache.

Example

```
var responseCache = new RedisNamespace("noderes");

var reply = function(error, result) {
    // completes http response
};

var miss = function () {
    // builds and caches http response
};

responseCache.getOrElse(JSON.stringify(query), reply, miss);
```

5.2 MDS

The MDS library is a convenience that maps closely onto the MDS REST API.

Importing

Simply import the mds module as follows:

```
var mds = require('../mds'),
```

Generic API Form

Each of the MDS API's takes the same functional form:

Argument	Type	Description
0	JSON Query Object	A query object that covers all common MDS parameters. See the relevant section below for further details
1	Callback Function	A function which will be called back to on response.

Argument	Type	Description
		See below for further details

JSON Query Object

Field	Type	Description
filter	JSON	A JSON object, that represents the MDS filter.
fields	JSON	A JSON array, that represents the fields requested.
limit	Int	An integer representing the size of the result set.
offset	Int	An integer representing the offset of the result set (how many records to skip).
sort	JSON	A JSON object that represents the MDS sort params.

Callback Function

```
function (error, response, body)
```

Field	Type	Description
error	Request Library Error Object	An error object as returned by the JavaScript 'request' library.
response	Request Library Response Object	A response object as returned by the JavaScript 'response' library.
body	Parsed JSON Response	A JSON object parsed from the MDS response.

Available API's

- mds.vod.editorials
- mds.vod.nodes
- mds.vod.promotions

- mds.vod.images
- mds.vod.series
- mds.vod.products
- mds.vod.version
- mds.btv.services
- mds.btv.programmes
- mds.btv.editorials
- mds.btv.products
- mds.btv.series
- mds.btv.version
- mds.offers.promotions

Example

```
var mds = require('../mds');

mds.vod.editorials({
    filter: {
        "voditem.nodeRefs": id
    },
    limit: 0
}, function(error, response, editorials) {
    if (error) {
        returnDirectCount(error);
    } else {
        var count = parseInt(editorials.total_records, 10);
        var ttl = maxAge(response);
        directCache.setWithExpiry(key, count, ttl);
        returnDirectCount(null, count);
    }
});
```

5.3 SDP

The SDP library is a convenience that maps closely onto the SDP REST API.

Importing

Simply import the sdp module as follows:

```
var sdp = require('../sdp')
```

Generic API Form

```
function (methodName, params, locale, token, callback)
```

JSON Query Object

Argument	Type	Description
methodName	String	Name of method on the service
params	JSON	JSON map of query parameters
locale	String	Locale of API
token	String	Token authentication for API
callback	Callback Function	See callback function below

Callback Function

```
function (error, response, body)
```

Field	Type	Description
error	Request Library Error Object	An error object as returned by the JavaScript 'request' library.
response	Request Library Response Object	A response object as returned by the JavaScript 'response' library.
body	Parsed JSON Response	A JSON object parsed from the MDS response.

Available API

- sdp.acquiredContentList

Example

```
sdp.acquiredContentListService("getByAccountUIDAndItemType", {arg0: query.accountUid,
 arg1: "CURRENT", arg2: "PKG"}, query.locale, query.token, function (error, response, acl
Items) {
   if (error) {
```

```
      error = error.response || error;
      return reject(getErrorObject("SDP", error.resultCode, error.result), null, res);
   }
   resolve(aclItems);
});
```

5.4 Third Party Node Modules

These are the third party node modules that are available for use in the cloud-application-framework:

async

1.5.2

https://www.npmjs.com/package/async

Async provides around 20 functions that include the usual 'functional' suspects (map, reduce, filter, each…) as well as some common patterns for asynchronous control flow (parallel, series, waterfall…). All these functions assume you follow the Node.js convention of providing a single callback as the last argument of your async function.

body-parser

1.14.2

https://www.npmjs.com/package/body-parser

Node.js body parsing middleware.

config

1.19.0

https://www.npmjs.com/package/config

Node-config organizes hierarchical configurations for your app deployments.

It lets you define a set of default parameters, and extend them for different deployment environments (development, qa, staging, production, etc.).

cors

2.7.1

https://www.npmjs.com/package/cors

CORS is a node.js package for providing a Connect/Express middleware that can be used to enable CORS with various options.

curry

1.2.0

https://www.npmjs.com/package/curry

A curry function without anything too clever (... because hunger is the finest spice)

express

4.13.4

https://www.npmjs.com/package/express

Fast, unopinionated, minimalist web framework

gm

1.21.1

https://www.npmjs.com/package/gm

GraphicsMagick and ImageMagick for node.js

image-size

0.4.0

https://www.npmjs.com/package/image-size

A Node module to get dimensions of any image file

logs4js

0.6.31

https://www.npmjs.com/package/log4js

This is a conversion of the log4js framework to work with node. I've mainly stripped out the browser-specific code and tidied up some of the javascript.

node-cache

3.1.0

https://www.npmjs.com/package/node-cache

Simple and fast NodeJS internal caching. Node internal in memory cache like memcached.

promise

7.1.1

https://www.npmjs.com/package/promise

This is a simple implementation of Promises. It is a super set of ES6 Promises designed to have readable, performant code and to provide just the extensions that are absolutely necessary for using promises today.

redis

2.4.2

https://www.npmjs.com/package/redis

This is a complete and feature rich Redis client for node.js. It supports all Redis commands and focuses on high performance.

request

2.69.0

https://www.npmjs.com/package/request

Request is designed to be the simplest way possible to make http calls. It supports HTTPS and follows redirects by default.

pm2

0.15.9

https://www.npmjs.com/package/pm2

PM2 is a production process manager for Node.js applications with a built-in load balancer. It allows you to keep applications alive forever, to reload them without downtime and to facilitate common system admin tasks.

pm2-logrotate

1.3.1

https://www.npmjs.com/package/pm2-logrotate

Module to rotate logs of every pm2 application

6 Bundled Plugins

6.1 Ping

Summary

The purpose of this API is to act as a simple test service to check that the cloud-application-framework is working correctly.

To use this API, send a HTTP GET request to the URL:

```
http://<host>/caf/ping
```

Response

```
HTTP 200
pong
```

6.2 Service Subscription

Summary

Currently, in order for the client application to retrieve BTV services and their associated subscription status it needs to make two separate requests - one to MDS to retrieve the services metadata and one to SDP to retrieve the acquired content list. The client then needs to process the two responses to flag which of the services are subscribed.

The purpose of the Services plugin is to expose an API that looks and behaves very similar to the MDS services request but can also return a subscription flag as part of the returned metadata.

Query Parameters

Name	Type	Description	Required
filter	JSON object	A key/value pairing list. Where key is a fieldname, and value is the value the field should be. If subscribed filter exists and is set to true, then only subscribed services will be return `filter={"regions":"Spain", "subscribed": true}`	No

Name	Type	Description	Required
sort	list of lists (pairs)	A list of pairs, where a pair is a list containing a field name and a sort order. Where a sort order of 1 is Ascending, and a sort order of -1 is descending. **`sort=[["Title",1]]`**	No
fields	list of strings	A list of fields names to return in the response. If "subscribed" exists, then an additional subscribed flag will be added to each service JSON object. **`fields=["subscribed", "Title", "regions", "parent", "ancestors", "children", "descendants"]`**	No
offset	integer	The number of records to skip. For pagination. **`offset=0`**	No
limit	integer	The number of records to return. For pagination. If subsc.ribed filter is set to true it is not certain that this many results will be returned **`limit=10`** **Warning!** Please note that limit=0 used to bring back unbounded results, but this behaviour has now been changed to return 0 results.	No
accountUid	string	Account Uid to be passed onto SDP Acquired Content request `accountUid="123456789"`	Yes
locale	boolean	Locale to be passed onto SDP Acquired Content request `locale="en_gb"`	Yes
token	string	Token to be passed onto SDP Acquired Content request `token="abcdefghijklmnopqrstuvwxyz"`	Yes

Response

HTTP 200 OK

The response will match that of an MDS btv/services request but if "subscribed" is included in the fields array parameter then each service JSON object that is returned will contain an additional subscribed flag

Name	Type	Description
`services`	JSON object	The services returned from MDS (each service will contain the additional subscribed flag if requested in the fields parameter)
`total_records`	int	Total number of services in response
`version`	String	MDS version

Examples

Make an MDS /vod/nodes request as normal:

```
/caf/btv/services?filter={"locale":"en_GB", "subscribed":
true}&fields=["subscribed"]&accountUid=1&locale=en_gb&token=123456789
```

and the response should show:

```
{
  "total_records": 2,
  "services": [
    {
      "technical": {
        "productRefs": [
          "LYS000000626"
        ]
      },
      "subscribed": true
    },
    {
      "technical": {
        "productRefs": [
          "LYS000000626"
        ]
      },
      "subscribed": true
    }
  ],
  "version": "20160225151223"
```

```
}
```

6.3 Node Count

Summary

There are two requirements that we have previously been given from client developers looking to fulfill their UI requirements and optimize their apps.

- Display the number of assets within a node whilst browsing nodes (i.e. before displaying the assets)
- Hide empty nodes, including parent nodes who's descendant nodes have no assets.

The architecture of the MDS and its underlying collection means that catalogue nodes and editorial data is kept separate. The content has references to node identifiers for linking, but not the other way around.

Therefore to fulfill the requirement the client must do multiple requests, one to get the nodes themselves, then a further editorial API request for each displayed node.

The purpose of this plugin is to enhance the MDS API in such as way as for these requirements to be satisfied and to allow the client to avoid having to make these multiple calls.

Query Parameters

Name	Type	Description	Required
filter	JSON object	A key/value pairing list. Where key is a fieldname, and value is the value the field should be. `filter={"regions":"Spain"}`	No
sort	list of lists (pairs)	A list of pairs, where a pair is a list containing a field name and a sort order. Where a sort order of 1 is Ascending, and a sort order of -1 is descending. `sort=[["Title",1]]`	No
fields	list of strings	A list of fields names to return in the response. `fields=["Title", "regions", "parent", "ancestors", "children", "descendants"]`	No
offset	integer	The number of records to skip. For pagination. `offset=0`	No

Name	Type	Description	Required
limit	integer	The number of records to return. For pagination. `limit=10` **Warning!** Please note that limit=0 used to bring back unbounded results, but this behaviour has now been changed to return 0 results.	No
hints	boolean	True to create pre-sorted "hints" to assist performance. False is the client believes there is no benefit in pre-sorting the data.	No
cache	boolean	True to allow caching on CDN, false to prevent CDN caching. This just sets or unsets HTTP headers.	No
pretty	boolean	True to return human readable formatted JSON. False to return compact and efficient JSON.	No

Response

HTTP 200 OK

An enhanced version of the MDS /vod/nodes API, with an additional field:

Name	Type	Description	Localized	Always available
`total_editorials`	int	The number of editorials existing under the given node	No	Yes

Examples

Make an MDS /vod/nodes request as normal:

```
/caf/[defined_endpoint]?pretty=true&limit=10&fields=["id","title","total_
editorials"]&filter={"isRoot":true, "region":"Sao Paulo"}
```

and the response should show with the enhanced field:

```
{
 "nodes": [
  {
      "id": "54347ccf-0425-46de-9a3c-180d3c635182",
        "title": "Recomenda",
   "total_editorials": 134
     },
     {
        "id": "662638c8-9683-48a8-b5a3-92678468d2dc",
        "title": "Lançamentos",
   "total_editorials": 56
     },
     {
        "id": "948cdcf3-0dce-4969-aa3c-6d315fbec2d5",
        "title": "Promoção de Hoje",
   "total_editorials": 29
     },
     {
        "id": "0d1c86d0-1af4-45d9-8927-2540f88358ea",
        "title": "Especial OSCAR 2014",
   "total_editorials": 78
     },
     {
        "id": "5301438f-5fe1-42d8-92ba-eee1ffd4af65",
        "title": "Especial Futebol",
   "total_editorials": 66
     },
     {
        "id": "1bc89c53-36c5-4ec6-9540-fd3834be8fc2",
        "title": "Ação/Ficção",
   "total_editorials": 44
     },
     {
        "id": "e25abc87-7e69-481f-afcb-5745d9cb4f57",
        "title": "Animação",
   "total_editorials": 73
     },
     {
        "id": "7a6c12d2-32f7-4f6f-8beb-66ddd2c84aeb",
        "title": "Aventura/Fantasia",
   "total_editorials": 35
     },
     {
        "id": "9422eb62-6aba-46d8-9da3-ab3ff93c6da8",
        "title": "Comédia",
   "total_editorials": 99
     },
     {
        "id": "69ff1da5-1b8f-461e-b913-3265b3844474",
        "title": "Drama"
```

```
    "total_editorials": 87
      }
    ],
  "total_records": 1222,
  "version": "20141011000313"
}
```

6.4 Image Transformer

Summary

The purpose of this API is to transform images for the client software on the fly. It allows the client to request images in a different size than the original, so even if the image does not exactly fit the required size, it can be stretched/skewed/shrunk and/or cropped to fit. Doing the re-size on the device can take up valuable processing time, whilst on the server side it can be done once and cached for a long time for other clients to use.

To use this API, send a HTTP GET request to the URL:

```
http://<host>:<port>/[sub-folder]/img/transform
```

Example

http://ssolab1.nagra.com/nbxdemo/img/transform?uri=http://nagr.tmsimg.com/v3/AllPhotos/30905/30905_v3_ba.jpg&w=200&h=400&stretch=true

Query parameters

Name	Type	Description	Required
uri	uri	The image URL in FULL.	**YES**
w	number	Required width in pixels	NO*
h	number	Required width in height	NO*
stretch	boolean	If true, then the image will be sized to the exact dimensions of the width and height asked for. If false, or not specified, the width and height become the maximum window size, and the image aspect ration is preserved. Cannot use in conjunction with crop.	NO

Name	Type	Description	Required
crop	boolean	If true, then the image will be sized to the exact dimensions of the width and height asked for. If false, or not specified, the width and height become the maximum window size, and the image aspect ration is preserved. Cannot use in conjunction with stretch.	

Caution!

*either h or w **must** be supplied. If only one is supplied, the aspect ratio is maintained, and stretch doesn't do anything.

Response

HTTP 200 - image

Usage

The Image transformation API is a dynamic API that internally spawns ImageMagick commands to process the image. Since ImageMagick must have access to the image on the local file system, the remote image is first downloaded to a source folder.

One downloaded to the source folder, the image is resized depending on the parameters supplied. The created image is then stored in a cache folder. The name of this new image depends on the parameters supplied. Finally, the cached image is sent back to the client.

For efficiency and performance, the cached folder is first checked for an image based on the image naming rules. If found, the cached image is returned. If a cached image is not found, the source folder is checked, as the original image might still be available locally (it could've been used to create a cached image with different attributes). If found, the source folder is resized and the result cached. Finally if neither cached or source image is found, the remote image is downloaded.

JavaScript Utility method

The below function transforms an image url to the transformed image url. It hardcodes the server address (could be substituted for configured url) and assumes you want to crop (since this usually gives the best results).

```
var transformImage = function(url, width, height) {
 return "http://ssolab1.nagra.com/nbxdemo/img/transform?uri=" + url + "&w=" + width +
 "&h=" + height + "&crop=true";
}
```

Then in the application, you simply swap e.g.:

```
doc.img.src = editorial.promoImage[0];
```

to

```
doc.img.src = transformImage(editorial.promoImages[0], 294, 165);
```

Examples

Case 1

For the movie "All is Lost" we have an original image measuring 540 x 720 px. The target image for a vod poster is 294 x 441 px.

This image is too big for the UI and the aspect ratio is off.

Transformation 1.1

```
/img/transform?uri=http://nagr.tmsimg.com/assets/p9936737_i_v4_aa.jpg&w=294&h=441
```

The image is now shrunk to an appropriate size, with the aspect ratio maintained. All the image detail is kept, but the image is slightly too narrow for the target width of 441 (its 392). The height is 294 as required.

Transformation 1.2

```
/img/transform?uri=http://nagr.tmsimg.com/assets/p9936737_i_v4_aa.
jpg&w=294&h=441&stretch=true
```

By stretching the image, we get the perfect size, the image is now 294 x 441. Because the aspect ratio is not exactly the same as the required size however, the image is squeezed a bit.

6.5 CMS4-to-NXgen

Summary

This module acts an adapter between CMS4 and the new (but not yet complete) CMS5 API, acting as an interim step.

The API's adapted are the following:

```
/rest/api/commercial/v1/contentAuthorizations?content.securityId=123&content.start
Offset=-P1M&content.endOffset=P1M&limit=100
```

Each API calls makes up to two API requests upstream to MDS, namely:

- First to look for LIVE products
- If no result, then to look for VOD products.

/contentAuthorizations

Request

This API allows a search for products related to content by drmId/securityId with an optional time period.

Name	Type	Description	Required
content.securityId	String	The security id of the related content.	Yes
content.start AvailabilityOffset	String [ISO8601 Duration]	The offset from now after which a content's availability should end.	Yes
content.end AvailabilityOffset	String [ISO8601 Duration]	The offset from now before which a content's availability should have started.	Yes
page	Number	The number of the page we want to retrieve	Only when limit is specified
limit	Number	The size of the page we want to retrieve	Only when page is specified

Response

Response matches the following RAML/XSD schema segements:

```
#%RAML 0.8
title: CPM contract for product retrieval in batch
baseUri: http://localhost:8380/rest/api/commercial/{version}
version: v1
documentation:
  - title: CPM API for product retrieval in batch
    content: REST API, definition of interface for product objects retrieval
schemas:
  - getContentAuthorization: !include getContentAuthorization.json
  - error: !include error.json

/contentAuthorizations:
  displayName: contentAuthorizations
  description: Handle contentAuthorization resources
  get:
    description: |
      Returns the list of products that sell a specific content for a given time period
    queryParameters:
      liveOptimized:
```

```
        description: |
          If the request is configured to be optimal for live, only live products are
 fetched otherwise a mix of live and VOD products are fetched.
        type: boolean
        required: false
        example: false
      content.securityId:
        description: |
          The API will return any product that sells either a technicalChannel or a
 technicalContent having a SecurityInfo.id with the given value.
          The way the product is related to the technicalChannel or technicalContent to
 sell is managed internally within the Content  and Product Manager.
          Mostly used during license authorization.
        type: string
        required: true
        example: 123
      content.startAvailabilityOffset:
        description: |
          The offset is expressed as a number of seconds. The window start date is now +
 offset now being the instant of the request. The API will return
          all the products that sell a content with an end validity greater than (or
 equal) the window start date. The end validity being either the product link
          (relationship entity between a product and the element sold) end date if
 provided, otherwise it is the technicalChannel or technicalContent end date
          that is used (depending which content the product sells).
          Mostly used during license authorization.
          Negative values are supported to define date prior to now.
          It must comply ISO 8601 standard ruled by the following pattern.
        type: string
        required: true
        example: P3Y6M4DT12H30M5S
        pattern: |
          [-]?P[0-9]{,4}[Y]?[0-9]{,2}[M]?[0-9]{,2}[D]?[0-9]{,2}[T]?[0-9]{,2}[H]?[0-9]{,2}
[M]?[0-9]{,2}[S]?
      content.endAvailabilityOffset:
        description: |
          The offset is expressed as a number of seconds. The window end date is now +
 offset now being the instant of the request. The API will return
          all the products that sell a content with an start validity lesser than (or
 equal) the window end date. The start validity being either the product link
          (relationship entity between a product and the element sold) start date if
 provided, otherwise it is the technicalChannel or technicalContent start date
          that is used (depending which content the product sells).
          Mostly used during license authorization.
          Negative values are supported to define date prior to now.
          It must comply ISO 8601 standard ruled by the following pattern.
        type: string
        required: true
        example: P1H
        pattern: |
```

```
          [-]?P[0-9]{,4}[Y]?[0-9]{,2}[M]?[0-9]{,2}[D]?[0-9]{,2}[T]?[0-9]{,2}[H]?[0-9]{,2}
[M]?[0-9]{,2}[S]?
    responses:
      200:
        description: returns the contentAuthorizations
        body:
          "application/json;charset=UTF-8":
            schema: getContentAuthorization
            example: !include ./examples/GetContentAuthorization.json
      500:
        description: Server internal error.
        body:
          "application/json;charset=UTF-8":
            schema: error
            example: |
              {
                "technicalMessage": "Wrong number of content found with content.security
Id",
                "businessRuleMessage": ""
              }
```

```
{
   "$schema": "http://json-schema.org/draft-04/schema#",
   "type": "object",
   "properties": {
      "content": {
         "$ref": "#/definitions/getContent"
      },
      "products": {
         "$ref": "#/definitions/getProducts"
      }
   },
   "required": [
      "products"
   ],
   "definitions": {
      "getContent": {
         "type": "object",
         "description": "Basic content properties necessary for the content authorization
 use case. In this context, content means resource identified by the content.securityId
 query parameter (e.g. technical content, technical channel ...)",
         "properties": {
            "id": {
               "description": "The content public id.",
               "type": "string"
            },
            "name": {
               "description": "The content name.",
               "type": "string"
```

```
            },
            "self": {
               "description": "The URI of the REST ressource.",
               "type": "string"
            },
            "rights": {
               "description": "Set of properties describing the consumption capabilities
of a content",
               "type": "object",
               "properties": {
                  "storageAllowed": {
                     "description": "Allow a client to download the Content",
                     "type": "boolean"
                  }
               }
            }
         },
         "required": [
            "id", "name", "self"
         ]
      },
      "getProducts": {
         "type": "object",
         "description": "Set of products with only information usefull for the content
authorization use case",
         "properties": {
            "totalRecords": {
               "description": "The number of products contained in the CPM matching the
query parameter filters.",
               "type": "integer",
               "minimum": 0
            },
            "productSet": {
               "description": "Set of products with only information usefull for the
content authorization use case",
               "type": "array",
               "items": {
                  "description": "Set of products with only information usefull for the
content authorization use cases",
                  "type": "object",
                  "properties": {
                     "id": {
                        "description": "The product public id. ",
                        "type": "string"
                     },
                     "name": {
                        "description": "The product name.",
                        "type": "string"
                     },
                     "self": {
                        "description": "The URI of the REST ressource.",
```

```
                        "type": "string"
                     },
                     "start": {
                        "description": "The product start date. ISO 8601 format up to the
 second and using the \"Z\" GMT time zone indicator. Example: 2012-05-31T06:10:30Z.",
                        "type": [
                           "string", "null"
                        ],
                        "pattern": "^([0-9][0-9][0-9][0-9]-[0-1][0-9]-[0-3][0-9]T[0-2]
[0-9]:[0-5][0-9]:[0-5][0-9]Z)$"
                     },
                     "end": {
                        "description": "The product end date. ISO 8601 format up to the
 second and using the \"Z\" GMT time zone indicator. Example: 2012-05-31T06:10:30Z.",
                        "type": [
                           "string", "null"
                        ],
                        "pattern": "^([0-9][0-9][0-9][0-9]-[0-1][0-9]-[0-3][0-9]T[0-2]
[0-9]:[0-5][0-9]:[0-5][0-9]Z)$"
                     },
                     "type": {
                        "description": "Enumeration allowing to differentiate products
 (allowed values: transactional, subscription)",
                        "type": "string",
                        "enum": [
                           "transactional", "subscription"
                        ]
                     },
                     "rights": {
                        "description": "Set of properties describing the consumption
 capabilities of a product",
                        "type": "object",
                        "properties": {
                           "impulsive": {
                              "description": "If true the product can be purchased by the
 end-user through the client application or can be entitled by the operator. If false the
 product can only be entitled by the opertor. If the value is not provided, every system
 will assume the value as false.",
                              "type": "boolean"
                           },
                           "rentalDuration": {
                              "description": "The amount of time during which the content
 sold by this product can be accessed after the start of the license (the start can be at
 purchase time in absolute mode or at first viewing time in relative mode). This applies
 only for transactional products. The format is based on ISO 8601 with restricted allowed
 pattern: PnDTnHnMnS. Must be positive.",
                              "type": "string"
                           },
                           "consumptionWindow": {
                              "description": "The amount of time after purchase during
 which initial access to the content sold by this product can be attempted, then the
```

```
 rentalDuration applies. When set this indicates Relative Mode usage rule calculations
 should apply. The field value indicates the range within which a relative Product
 viewing can commence. A license calculation agent determines the final end time of
 Product viewing based on rental duration and its own configuration. When this value is
 omitted it means. This applies only for transactional products. The format is based on
 ISO 8601 with restricted allowed pattern: PnDTnHnMnS. Must be positive.",
                              "type": "string"
                           },
                           "storageAllowed": {
                              "description": "The Product can be marked as downloadable
 setting this flag to true, allowing Content also marked as downloadable (with the same
 flag set to true) to be downloaded to a Client. Where Product is not marked as such then
 any associated Content will not be downloadable whatever the Content status.",
                              "type": "boolean"
                           }
                        }
                     }
                  },
                  "required": [
                     "id", "name", "self", "type"
                  ]
               }
            }
         },
         "required": [
            "totalRecords"
         ]
      }
   }
}
```

Example

```
/rest/api/commercial/v1/contentAuthorizations?content.securityId=drm1&content.start
AvailabilityOffset=-P1Y&content.endAvailabilityOffset=P1Y&limit=1&page=2
```

```
{
   "content": {
      "id": "contentId1",
      "name": "content name 01",
      "self": "http://cpmhost:8080/rest/api/content/v1/technicalContents/contentId1",
      "rights": {
         "isStorageAllowed": false
      }
   },
   "products": {
      "totalRecords": 4,
      "productSet": [
         {
```

```
            "id": "ppid01_pprid01",
            "name": "product name 01",
            "self": "http://cpmhost:8080/rest/api/commercial/v1/products/ppid01_pprid01",
            "start": "2015-01-01T00:00:00Z",
            "end": "2015-02-01T00:00:00Z",
            "type": "transactional",
            "rights": {
               "impulsive": true,
               "rentalDuration": "P4DT12H30M5S",
               "consumptionWindow": "P2DT12H30M5S",
               "isStorageAllowed": true
            }
         }, {
            "id": "ppid01_pprid02",
            "name": "product name 02",
            "self": "http://cpmhost:8080/rest/api/commercial/v1/products/ppid01_pprid02",
            "type": "subscription",
            "start": "2015-01-01T00:00:00Z",
            "end": null,
            "rights": {
               "impulsive": false
            }
         }, {
            "id": "ppid02_pprid02",
            "name": "product name 03",
            "self": "http://cpmhost:8080/rest/api/commercial/v1/products/ppid02_pprid02",
            "start": "2015-01-01T00:00:00Z",
            "end": "2015-02-01T00:00:00Z",
            "type": "transactional",
            "rights": {
               "impulsive": true,
               "rentalDuration": "P4DT12H30M5S",
               "consumptionWindow": "P2DT12H30M5S",
               "isStorageAllowed": false
            }
         }, {
            "id": "ppid02_pprid01",
            "name": "product name 04",
            "self": "http://cpmhost:8080/rest/api/commercial/v1/products/ppid02_pprid01",
            "type": "subscription",
            "start": null,
            "end": null,
            "rights": {
               "impulsive": false
            }
         }
      ]
   }
}
```

6.6 VoD Series-Product Aggregation

Summary

This plugin enables the enhancement of the MDS /vod/series API to add aggregated products that apply to a given series. The API is fully functional as per the MDS /vod/series specification.

To enable the plugin just add a mapping such as the following to the CAF Registry configuration:

```
{ "/vod/series":"./lib/series-products-join" }
```

It will then be available at the endpoint **`/caf/vod/series.`**

Query Parameters

Name	Type	Description	Required
filter	JSON object	A key/value pairing list. Where key is a fieldname, and value is the value the field should be. **`filter={"id":"xyz"}`**	No
sort	list of lists (pairs)	A list of pairs, where a pair is a list containing a field name and a sort order. Where a sort order of 1 is Ascending, and a sort order of -1 is descending. **`sort=[["Title",1]]`**	No
fields	list of strings	A list of fields names to return in the response. **`fields=["Title"]`**	No
offset	integer	The number of records to skip. For pagination. **`offset=0`**	No
limit	integer	The number of records to return. For pagination. **`limit=10`** **Warning!** Please note that limit=0 used to bring back unbounded results, but this	No

Name	Type	Description	Required
		behaviour has now been changed to return 0 results.	
hints	boolean	True to create pre-sorted "hints" to assist performance. False is the client believes there is no benefit in pre-sorting the data.	No
cache	boolean	True to allow caching on CDN, false to prevent CDN caching. This just sets or unsets HTTP headers.	No
pretty	boolean	True to return human readable formatted JSON. False to return compact and efficient JSON.	No

Response

HTTP 200 OK

An enhanced version of the MDS /vod/nodes API, with an additional field:

Name	Type	Description	Localized	Always available
`products`	List<Product>	A list of products related to this series through associated content	Yes	Yes

Examples

Make an MDS /vod/seriesrequest as normal:

```
/caf/vod/series?filter={"id":"269417"}
```

and the response should show with the enhanced field:

```
{
  "series": [
    {
      "Rating": {
        "code": "0",
```

```
        "precedence": 0,
        "Title": "Label for 0"
      },
      "nls": {
        "Title": "4b3d372d2f0109018f8f858f8f00"
      },
      "Title": "SLiDE",
      "locale": "en_AU",
      "title": "SLiDE/269417",
      "deviceType": [],
      "id": "269417",
      "products": [
        {
          "startPurchase": 1388534400,
          "endValidity": 1577836799,
          "title": "CCA_PDL_BXS",
          "price": {
            "startPurchase": 1388534400,
            "billingInterval": "1",
            "endPurchase": 1577836799,
            "value": 0,
            "currency": "AUD",
            "subscriptionDurationRatio": "1",
            "billingTimeUnit": "month"
          },
          "startValidity": 1388534400,
          "endPurchase": 1577836799,
          "deviceType": [],
          "type": "subscription",
          "id": "CCA_PDL_BXS"
        }
      ]
    }
  ],
  "version": "20170104105325",
  "total_records": 1
}
```


6.7 Caching Channel Facade *New*

Summary

This lightweight decorator is a caching facade to the SDP channel service. Calls are cached in Redis for a configurable amount of time to avoid expensive delegation to SDP.

Architecture

The cache is meant to enhance the start-up routine of a STB, as per the following diagram:

Redis Configuration

This module is dependent upon a Redis sentinel setup to operate correctly. If using puppet, please add the **cloud-application-framework::caf-redis** class to all nodes in your Redis cluster:

Denote one of your cluster as the initial master. Please note that the master will change as Redis fails over, this is just a preliminary setting.

CAF Configuration

For the standard **cloud-application-framework::cloud-application-framework** class, ideally you should configure the following values:

These include:

	Value	Description
sdpPath	`/hue-gateway/gateway/http/js/`	The base path for the SDP hue services.
ensureversion	`<your_version>`	The version of the CAF you want installed.
sdpUrl	`http://127.0.0.1:8180`	Reference to the local SDP instance. Please ensure this is between nginx and SDP, not outside of nginx, otherwise token semantics may cause issues.
redis_sentinels	`172.16.1.64:26379`	A comma-list of host:port redis sentinel members.
registry	`{ "/admin/v1/ping": "./lib/ping", "/ping": "./lib/ping", "/hue-gateway/gateway/http/js/channelService":`	Make sure you add the channel service to your module registry.

	Value	Description
	"./lib/channel-service-facade" }	

Nginx Configuration

We can transparently deploy the CAF in between nginx and SDP without disturbing any consuming clients.

In the CAF deployment you will find two sample files to help you enable this configuration:

- **/opt/nginx/conf/http/channel-service-facade.conf.snippet**
- **/opt/nginx/conf/server_80/channel-service-facade.conf.snippet**

The former should be added to the nginx file, **/opt/nginx/conf/http/local.conf**. It contains the following:

```
upstream caf_sdp_override {
    server 127.0.0.1:3000;
    server 127.0.0.1:8180 backup;
    keepalive 50;
}
```

This will route requests primarily to CAF, but if for some reason CAF cannot serve requests it will fall back to the normal SDP route.

The second file should be added to, /opt/nginx/conf/server_80/local.conf. It contains a location handler to override the default in SDP:

```
location /hue-gateway/gateway/http/js/channelService/getAllAuthorizedCCLSForDevice {
    include cors.conf;
    rewrite_by_lua_file /opt/nginx/lib/lua/token.lua;
    allow all;
    proxy_pass http://caf_sdp_override;
}


location /hue-gateway/gateway/http/js/channelService/refreshCCL {
    include cors.conf;
    allow 127.0.0.1;
    allow 172.0.0.0/8;
    deny all;
    proxy_pass http://caf_sdp_override;
}
```

Caution! These have not been enabled by default to give operations more control over when this feature is enabled, rather than just doing so on puppet installation.

Caution! Certain allow rules have been created on some of the admin API's. Please change the CIDR block to allow your own subnet.

API

```
openapi: 3.0.1

info:
  title: Caching Channel Service Facade
  description: Facade over SDP channel service that caches responses for performance
  version: 1.0.0

servers:
- url: https://cloud-application-framework

paths:

  /hue-gateway/gateway/http/js/channelService/getAllAuthorizedCCLSForDevice:

    get:
      tags:
      - Channel Service
      summary: Get authorized channel call numbers for device
      description: Get authorized channel call numbers for device
      parameters:
      - name: token
        in: header
        description: SDP token for authentication
        required: true
        schema:
          type: string
      responses:
        200:
          description: successful operation
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/AuthorizedCCLSResponse'

    post:
```

```
      tags:
      - Channel Service
      summary: Get authorized channel call numbers for device
      description: Get authorized channel call numbers for device
      parameters:
      - name: token
        in: header
        description: SDP token for authentication
        required: true
        schema:
          type: string
      responses:
        200:
          description: successful operation
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/AuthorizedCCLSResponse'

  /hue-gateway/gateway/http/js/channelService/refreshCCL:

    post:
      tags:
        - Admin
      summary: Refresh all content cached in Redis
      description: Refresh all content cached in Redis
      responses:
        200:
          description: Refresh operation started ok

    get:
      tags:
        - Admin
      summary: Get current status of any refresh process
      description: Get current status of any refresh process
      responses:
        200:
          description: Refresh details returned ok

    delete:
      tags:
        - Admin
      summary: Cancel refresh process
      description: Cancel refresh process
      responses:
        200:
          description: Refresh operation cancelled


components:
```

```
  schemas:

    AuthorizedCCLSResponse:
      type: object
      properties:
        result:
          type: array
          items:
            type: string
        requestId:
          type: integer
        resultCode:
          type: string
```

7 Best Practices

Warning!

These are the best practices to be followed by any developer of a cloud-application-framework plugin.

It is important to pay close attention to these practices during development and maintenance of a plugin for the following reasons:

- Convention allows for simpler maintenance and debugging
- Performance is of extreme importance, as your code may be deployed in a production environment alongside many other medialive head-end modules, all contending for resource. A badly written module could have very real consequences on the performance of the platform as a whole.

1. Follow the standard node.js function conventions

Callback convention

Modules should expose an error-first callback interface, for example:

```
module.exports = function(name, callback) {
 var account = createAccount(name);
    // note, that the first parameter is the error
    // which is null here
    // but if an error occurs, then a new Error
    // should be passed here
    return callback(null, account)
}
```

Always check for errors in callbacks

To better understand why this is a must, first start with an example that is **broken** in every possible way, then fix it.

```
var fs = require('fs');

function readJSON(filePath, callback) {
    fs.readFile(filePath, function(err, data) {
        callback(JSON.parse(data));
    });
}

readJSON('./package.json', function (err, pkg) { ... }
```

The very first problem with this **`readJSON`** function, is that it never checks, if an **`Error`** happened during the execution. You should always check for them.

The improved version:

```
function readJSON(filePath, callback) {
  fs.readFile(filePath, function(err, data) {
```

```
    // here we check, if an error happened
    if (err) {
      // yep, pass the error to the callback
      // remember: error-first callbacks
      callback(err);
    }

    // no error, pass a null and the JSON
    callback(null, JSON.parse(data));
  });
}
```

Return on callbacks

One of the problems that still exists in the above example, is that if an **`Error`** occurs, then the execution will not stop in the **`if`** statement, but will continue. This can lead to lots of unexpected things. As of a rule of thumb, always return on callbacks.

```
function readJSON(filePath, callback) {
  fs.readFile(filePath, function(err, data) {
    if (err) {
      return callback(err);
    }
    return callback(null, JSON.parse(data));
  });
}
```

2. Use ample logging

Please include logging statements in your code at all the relevant points to make production debugging and monitoring simpler.

Whilst logging, carefully consider the level (debug, info, warn, error) at which you log to allow the administrator more fine grained control over what they want to see.

3. Handle errors properly

Node apps crash when they encounter an uncaught exception. Not handling exceptions and taking appropriate actions will make your Express app crash

To ensure you handle all exceptions:

- always check the error response of a function for the presence of an error
- use try-catches for the limited synchronous code you write
- take advantage of promises for cleaner handling of errors

Use try catch

Try-catch is a JavaScript language construct that you can use to catch exceptions in synchronous code. Use try-catch, for example, to handle JSON parsing errors, as shown below:

```
app.get('/search', function (req, res) {
  // Simulating async operation
```

```
  setImmediate(function () {
    var jsonStr = req.query.params;
    try {
      var jsonObj = JSON.parse(jsonStr);
      res.send('Success');
    } catch (e) {
      res.status(400).send('Invalid JSON string');
    }
  });
});
```

Use promises

Promises will handle any exceptions (both explicit and implicit) in asynchronous code blocks that use `then()`. Just add `.catch(next)` to the end of promise chains. For example:

```
app.get('/', function (req, res, next) {
  queryDb()
    .then(function (data) {
      // handle data
      return makeCsv(data)
    })
    .then(function (csv) {
      // handle csv
    })
    .catch(next);
});
```

4. Never make blocking calls

Blocking Calls

Synchronous functions and methods tie up the executing process until they return. A single call to a synchronous function might return in a few microseconds or milliseconds, however in high-traffic websites, these calls add up and reduce the performance of the app. Avoid their use in production.

Although Node and many modules provide synchronous and asynchronous versions of their functions, always use the asynchronous version in production. The only time when a synchronous function can be justified is upon initial startup.

CPU Intensive Functions

You must also be aware that code which requires a lot of CPU (and so takes significant time to process) has exactly the same effect as code blocked by I/O. Make sure that your code is efficient and, in the worst case if some intensive calculation is required, use memoization or split the calculation up into many discrete stages to free up the thread in use. In other words, don't start calculating the digits to PI to some arbitrary length, or nested for loops with a combinatoric explosion of steps.

5. Cache whenever possible

Cache any shared state

Many calls may be either completely or at least partially shared between different users, and therefore across different API requests.

For example, take the case where we may want to enhance MDS nodes with the number of items existing within that node, so that we can filter out any empty catalogue entries. As these requests are not unique to any one user, we can happily cache the response and avoid many expensive back-end calls on the majority of the requests.

```
var reply = function(error, result) {
    if (error) {
        res.set(500).send(error);
    } else {
        res.send(result);
    }
};

var miss = function(error, result) {
 mds.vod.nodes(query, function (error, response, nodes) {
        if (error) {
         return reply(error);
     }
        var extended = extendNodesWithCount(query, nodes);
     cache.set(query, extended)
 });
}

cache.getOrElse(query, reply, miss);
```

Use ttl's to provide automated cache expiry

As cached items are unlikely to be valid until the end of time, we need to purge them at some point to avoid stale data.

Rather than working out some complicated purge process, you can simply specify a ttl (the number of seconds until expiry) upfront, and the cache will automatically purge after this time.

```
cache.setWithExpiry(query, extended, ttl)
```

6. Play well with reverse proxies

Reverse proxies, such as Nginx or any CDN are able to read the headers in your HTTP response and cache the response appropriately for a period of time. This prevents any subsequent calls to the same API even hitting the cloud-application-framework, allowing us to handle much more load than would otherwise be possible.

Please consider setting headers in your response / handling request headers that facilitate this function, more specifically:

Cache-Control: max-age

This directive specifies the maximum time in seconds that the fetched response is allowed to be reused for from the time of the request - e.g. “max-age=60” indicates that the response can be cached and reused for the next 60 seconds.

Last-Modified

The Last-Modified entity-header field value is often used as a cache validator. In simple terms, a cache entry is considered to be valid if the entity has not been modified since the Last-Modified value.

If-Modified-Since

The If-Modified-Since request-header field is used with a method to make it conditional: if the requested variant has not been modified since the time specified in this field, an entity will not be returned from the server; instead, a 304 (not modified) response will be returned without any message-body.

7. Use good async patterns

It is easy to make your life difficult by having chains of callbacks many levels deep. This anti-pattern will likely lead to un-maintainable, bug-prone, inflexible code. Please use libraries to avoid this anti-pattern, such as the two below:

async

Async is a utility module which provides straight-forward, powerful functions for working with asynchronous JavaScript.

Async provides around 20 functions that include the usual 'functional' suspects (**`map`**, **`reduce`**, **`filter`**, **`each`**…) as well as some common patterns for asynchronous control flow (**`parallel`**, **`series`**, **`waterfall`**…). All these functions assume you follow the Node.js convention of providing a single callback as the last argument of your **`async`** function.

promises

Promises help you naturally handle errors, and write cleaner code by not having callback parameters, and without modifying the underlying architecture (i.e. you can implement them in pure JavaScript and use them to wrap existing asynchronous operations).

The core idea behind promises is that a promise represents the result of an asynchronous operation. A promise is in one of three different states:

- pending - The initial state of a promise.
- fulfilled - The state of a promise representing a successful operation.
- rejected - The state of a promise representing a failed operation.

Once a promise is fulfilled or rejected, it is immutable (i.e. it can never change again).

8. Performance test

Lastly, always performance test any change you make to your plugin to ensure that you haven't adversely affected it.

Some performance testing tools we commonly use include:

- wrk
- gatling
- apache-bench

8 Implementing a CAF Module

Introduction

This document aims to provide step by step instructions on how to write and then integrate a node module into the Cloud Application Framework (CAF). It will walk through an example of creating a version endpoint that will simply return the version of the CAF back to the client.

Add Endpoint via Express Router

The Javascript code for each CAF module is located in /<ROOT>/lib directory of the CAF source code build.

1. To add a new endpoint, firstly create a new folder in the lib directory. For our version example we will add a folder called "version".

2. Add an index.js file to your folder.

3. Load express module, create a handler for GET requests (or PUT/POST/DELETE) and export module.

```
var express = require('express');
var router = express.Router();

router.get('/', function (req, res) {
});

module.exports = router;
```

4. Add logging and send response to client

```
var express = require('express');
var router = express.Router();

var logger = require('../logging').logger("version");
var packageJSON = require('../../package.json');

router.get('/', function (req, res) {
 var version = packageJSON.version;
 logger.debug('CAF version: ' + version);
 res.send(version);
});

module.exports = router;
```

5. Configure and enable the endpoint in config/default.json file

```
"Registry": {
    "/img": "./lib/image-transformer",
    "/btv": "./lib/btv",
    "/ping": "./lib/ping",
    "/version": "./lib/version"
},
```

6. Test the new endpoint by firstly starting node server with command "pm2 start caf.js -i 0" (you may need to install pm2 using command "npm install pm2 -g") and then hitting your endpoint from a browser

Writing and Running Unit Tests

We recommend that you use Mocha to write your unit test as there is a grunt task configured to run mocha tests and provide code coverage for these tests.

1. Create a test directory in your lib folder an add <MY_LIB>Test.js file

2. Write unit test

```
var requireHelper = require('../../../test/requireHelper'),
    request = require('supertest'),
    packageJSONVersion = require('../../../package.json').version,
    app = requireHelper("lib/app"),
    expect = require("chai").expect;
describe("Version Tests", function() {
    describe("Test Version GET API", function() {
        it("Responds with the correct version", function(done) {
             request(app)
                .get('/caf/version')
                .expect(200)
                .end(function(err, result) {
                    expect(result.text).to.equal(packageJSONVersion);
                    done();
                });
        });
    });
});
```

3. Run tests via grunt test task

Generate API Docs

We recommend that you use APIDocs to write your unit test as there is a grunt task configured to generate APIDoc documentation. For further information on APIDoc please see http://apidocjs.com/

1. Add API annotaions to library source code

```
/**
 * @api {get} /version
 * @apiName getVersion
 * @apiGroup version
 * @apiSuccess {String} Version of the Cloud Application Framework
 * @apiExample Example usage:
 *  http://node_domain:port/modules/path/caf/version
 */
router.get('/', function (req, res) {
 var version = require('../../package.json').version;
 logger.debug('CAF version: ' + version);
 res.send(version);
});
```

2. Run grunt docs task and documentation will be generated in <ROOT>/docs directory

9 Foxtel

9.1 Admin APIs RAML

```
#%RAML 1.0
title: CAF_Foxtel
version: v1
baseUri: http://caf/purchase/admin/purchases
mediaType:  application/json


/cache:
    delete:
      description: Delete the entire contents of the cache

      responses:
        200:
          body:
            application/json:

               example:
                {
                }
        401:
          description: Unauthorized
          body:
             application/json:
               example:
                {
                }

/cache/{accountNumber}:
    delete:
      description: Delete the cache value for an account.

      responses:
        200:
          body:
            application/json:
               example:
                {
                  deletedCount: <numberOfEntriesDeleted>
                }
        401:
          description: Unauthorized
          body:
             application/json:
```

```
                example:
                 {
                 }
```

9.2 Circuit Breaker

In 1.0.7 a circuit breaker has been introduced to cause requests to fail early if a service is constantly failing. The package being used is https://www.npmjs.com/package/opossum.

Services

Currently the circuitbreaker exists between the following services:

- Redis (for both medialive and federated layers)
- Purchase server (medalive layer)
- SDP (medialive layer)
- MDS (medialive layer)

Options

There are various options that may be set in the config file (or puppet before deployment):

- enabled (boolean): Set to true to enable the circuit breaking behaviour
- options.timeout (integer): Minimum time for a request to be considered timed out (circuit breaker will cancel it and consider it a failure). Note that this is one request from CAF to one of the services mentioned above and not the entire transaction between the client and CAF.
- options.errorThresholdPercentage (integer): Percentage of requests that fail before the circuit breaker opens for that service.
- options.resetTimeout (integer): Time taken for a circuit breaker to switch from "open" to "half-open" and to attempt to re-establish that service.

9.3 Client APIs RAML

```
#%RAML 1.0
title: CAF_Foxtel
version: v1
baseUri: http://caf/purchase
mediaType:  application/json


/purchases/product/{productId}:
  post:
    description: Make a purchase using productId
    headers:
      accountNumber: string
```

```
      mpDeviceId: string

    responses:
      200:
        body:
          application/json:

            example:
                {
                  "successResponse": {
                   "expiryDate": 1505488690000,
                   "creationDate": 1502811248000,
                   "consumptionWindowSecs": 172800,
                   "purchaseType" : "IMPULSE",
                   "firstAuthDate" : 1502811411000,
                   "productID" : "CMS123",
                   "contentDrmId" : "123",
                   "contentTitle" : "Star Wars"
                 }
              }

      404:
        description: Not Found
        body:
          application/json:
            example:
              {
              }


/purchases/contentDrm/{contentDrmId}:
  get:
    description: Checks if the account is authorised to play the content
    headers:
      accountNumber: string
      mpDeviceId: string

    responses:
      200:
        body:
          application/json:
            example:
              {
                "successResponse": {
                "expiryDate": 1505488690000,
                "creationDate": 1502811248000,
                "consumptionWindowSecs": 172800,
                "purchaseType" : "IMPULSE",
                "firstAuthDate" : 1502811411000,
                "productID" : "CMS123",
                "contentDrmId" : "123",
```

```
                "contentTitle" : "The Power of Pi"
                }
            }
      404:
        description: Not Found
        body:
          application/json:
            example:
              {
              }

/purchases:
    get:
      description: Retrieve all purchases for the account
      headers:
        accountNumber: string
        mpDeviceId: string

      responses:
        200:
          body:
            application/json:

               example:
                 {
                       "successResponse": {
                         "purchase" : [
                           {
                             "expiryDate": 1505488690000,
                             "creationDate": 1502811248000,
                             "consumptionWindowSecs": 172800,
                             "purchaseType" : "IMPULSE",
                             "firstAuthDate" : 1502811411000,
                             "productID" : "CMS123",
                             "contentDrmId" : "123",
                             "contentTitle" : "A film made by me"
                           },
                           {
                             "expiryDate": 1505488690000,
                             "creationDate": 1502811248000,
                             "consumptionWindowSecs": 172800,
                             "purchaseType" : "IMPULSE",
                             "firstAuthDate" : 1502811411000,
                             "productID" : "CMS124",
                             "contentDrmId" : "124",
                             "contentTitle" : "Another film made by me - the Sequel"
                           }
                         ]
                       }
                     }
        404:
```

```
          description: Not Found
          body:
             application/json:
               example:
                {
                }
```

9.4 Current State of Release 1.0.8 *New*

This update brings:

- Fix of bug where the MDS calls are missing a required field.

9.5 HTTP Router

Whitelisting

List of IP's can be allowed and denied access to CAF purchase API's without requiring an SDP token using puppet parameters foxtel_nginx_allow_api and foxtel_nginx_deny_api.

Auth Token

Clear text token and encoded Client-Token Token are auto generated using the accountNumber, mpDeviceId and accountUid (extracted from sdp using accountNumber) when purchase API's are invoked without having to supply a valid STB signon token.

Client-Token is used to make calls to sdp and clear text token is used to make calls to purchase server.

9.6 Puppet deployment

- In puppet, edit the registry parameter and add the following to the JSON object: "/purchase/purchases":"./lib/foxtel Purchase" and "/purchase/admin/purchases":"./lib/foxtelPurchaseAdmin"
- Also set the foxtelLayer parameter to either "FEDERATED" or "MEDIALIVE" depending on the CAF's deployment.
- Set the parameters for the redis host and port to point at your deployed redis.
- On the FEDERATED CAF, change the parameter foxtelMedialiveCafUrl to the address of the medialive nginx server.
- On the MEDIALIVE CAF, change the parameters for the purchase server endpoint, sdp url, and mds endpoint to the correct hosts and ports.
- For enabling HTTP Router whitelisting change parameter caf_foxtel_nginx_layer to Federated and add update parameters foxtel_nginx_allow_api and foxtel_nginx_deny_api to allow and deny IP's.
- For enabling token creation change parameter caf_foxtel_nginx_layer to Medialive
- For using admin APIs the list of allowed ips should be added to parameter caf_foxtel_admin_ip_list

9.7 Purchase APIs

9.7.1 DELETE - /caf/purchase/admin/purchases/cache

/caf/purchase/admin/purchases/cache

DELETE

Delete the entire contents of the cache

Response Messages

HTTP Status Code	Reason	Response Model
200	OK	
401	Unauthorized	

9.7.2 DELETE - /caf/purchase/admin/purchases/cache/<accountNumber>

/caf/purchase/admin/purchases/cache/<accountNumber>

DELETE

Delete the cache value for an account.

Response Messages

HTTP Status Code	Reason	Response Model
200	OK	{deletedCount: <numberOfEntries Deleted>}
401	Unauthorized	

9.7.3 GET - /caf/purchase/purchases

/caf/purchase/purchases

GET

Get all purchases for an account

Parameters

Parameter	Description	Parameter Type	Data Type	Required
accountNumber	The accountNumber for the purchases	header	string	Y
mpDeviceId	The id for the device	header	string	Y

Response Messages

HTTP Status Code	Reason	Response Model
200	OK	JSON - PurchaseList Response
401	Unauthorized	

9.7.4 GET - /caf/purchase/purchases/contentDrm/<contentDrmId>

/caf/purchase/purchases/contentDrm/<contentDrmId>

GET

Get purchase by contentDrmId, ensuring that the purchase is authorized to be played

Parameters

Parameter	Description	Parameter Type	Data Type	Required
accountNumber	The accountNumber for the purchases	header	string	Y
mpDeviceId	The id for the device	header	string	Y
contentDrmId	The id for the contentDrm of the purchase you wish to get	path	string	Y

Response Messages

HTTP Status Code	Reason	Response Model
200	OK	JSON - Purchase Response
401	Unauthorized	

9.7.5 POST - /caf/purchase/purchases/product/<productId>

/caf/purchase/purchases/product/<productId>

POST

Make a purchase of the specified productId.

Parameters

Parameter	Description	Parameter Type	Data Type	Required
accountNumber	The accountNumber for the purchases	header	string	Y
mpDeviceId	The id for the device	header	string	Y
productId	The id for the productId that the user wishes to purchase	path	string	Y

Response Messages

HTTP Status Code	Reason	Response Model
201	Created	JSON - Purchase Response
401	Unauthorized	