Kylone MicroCMS
Endpoint API for Android User Guide

Version 2.2.0

Revision Jan 18, 2019

CONTENTS

1. Introduction

Kylone provides an Endpoint API for developers to help creating a custom application to use with Kylone MicroCMS server including accessing and utilizing several types of content related data located on server and ready to use methods while working with the server.

Beside accessing and using such data easily while doing development of custom endpoint applications, the main purpose of the API is to maintain functionalities on both server and endpoint side and to be sure about the consistency of whole framework.

Kylone Endpoint API for Android will support following platforms;

Starting from MicroCMS v2.2.0	Starting from Android API-19 (4.4.2), armeabi-v7a only

This document includes guidelines for connecting and fetching data from server, utilizing content related data and some guidelines about usage of functionalities on server side to maintain same functionalities on endpoint side. Therefore, the API and this document will not cover topics like user specific tasks should be taken (such as checking network connectivity before trying to connect to target host, running on WiFi or Ethernet based networks only) or user or platform specific software development tasks (such as developing user interface or media players). Instead, provides ready to use methods for collecting components from server and organizing them to use easily with custom application.

2. Using the Endpoint API

There are two main components of the Endpoint API for Android;

- `src/com/kylone/shcapi/shApiMain.java`

  Single Java class which has several data-types as subclasses and some methods which should be overridden.

- `lib/cLshcapi.so`

  JNI implementation (native interface library) working conjunction with Java class `shApiMain`.

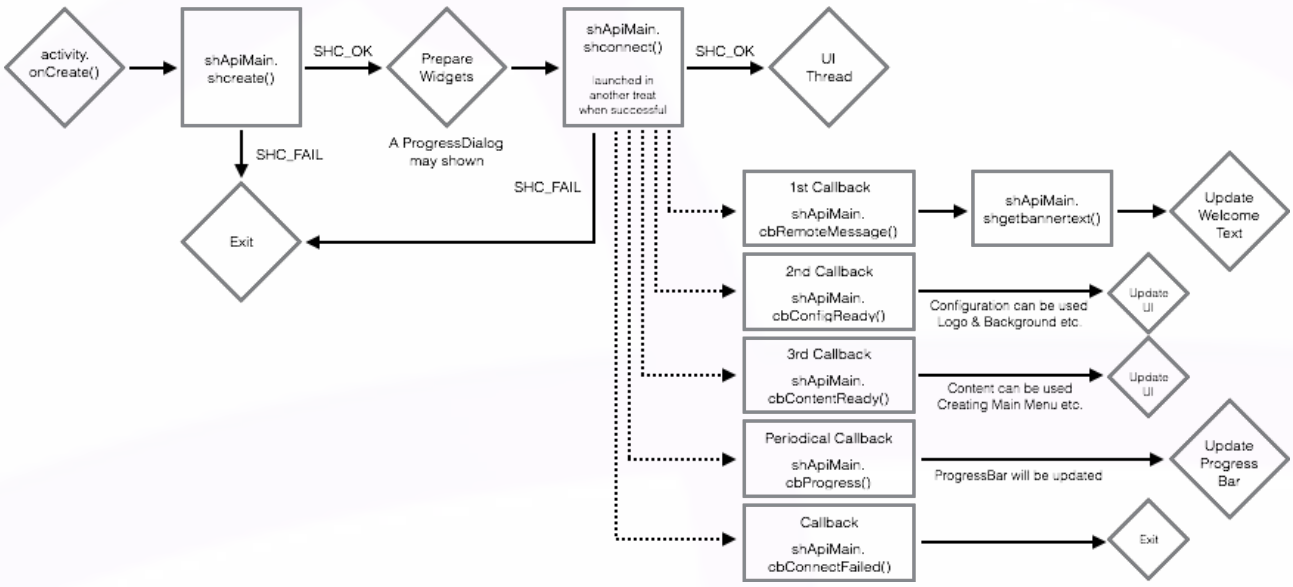
Custom applications will use the API by extending `shApiMain` class and overriding some of its methods. Basically, the main Java class consists of following items;

- Static methods exported by native interface
- Call-back methods which should be overridden and implemented by the custom application
- Definitions of some constants and variables
- Structured data types which implemented as class and it's methods.
- Some helper methods statically implemented to use by custom application

2.1. Workflow

Static methods exported from native interface should be used in a workflow shown below. All other helper functions can be used instantly whenever they are required.

In order to start using API, the essential method "`shcreate()`" should be called first. This method will return `SHC_OK` once successful. The second essential method is the "shconnect()". This method starts running the API and triggers required callback functions accordingly.



2.2. Main API Calls

```
native static int shcreate(String cachedir);
```

This method will prepare environment and starts threads inside the API. It will called with single option which for setting the cache directory inside the API. The most common usage of this method is shown below;

```
if (shApi.shcreate(getFilesDir().getAbsolutePath()) != shApi.SHC_OK) {
   feedback("! onCreate(): Failed to create API\n");
   return;
};
```

When `SHC_OPT_DOCACHE` option provided while calling `shconnect()` method, there will be a new folder called `".shcache"` created under given folder. This `".shcache"` folder should be exists while the application running. It can be freely deleted if caching will not be used or while termination of the application.

```
native static int shconnect(String targethost, int cache);
```

Custom application will start communicating with the server when calling this method.

It will be good idea to check network connectivity before calling this method. There are a few static methods implemented in the API for such purpose, please check `wifiConnected()` and `inetConnected()` helper methods explained in further sections.

There are two options needs to be provided while calling the method: IP address or domain name of the server and the caching option. The most common usage of the method is shown below;

```
if (shApi.shconnect(targetHost, shApi.SHC_OPT_DOCACHE) != shApi.SHC_OK) {
   Log.e("MainActivity", "shconnect(): Failed to connect target host\n");
```

```
};
```

Even if the method return immediately with `SHC_OK` code, it will create a new thread and keeps running inside it till `cbContentReady()` triggered. Required callback methods explained next section should be implemented to handle such tasks by the custom application.

```
native static int shserverisready(String targethost, int timeout)
```

This native method checks server status by performing tests in application layer and it can be used to see whether the server is functioning or not. Custom application may perform this test before trying to connect to server and it may give up after doing few more tests. For example, server may be doing boot at that time. On the other hand, this test will give more control to understand issues while doing troubleshooting etc.

Custom application may use this method before calling `shconnect()` method. The method returns `SHC_OK` upon successful and the server is ready to connect, otherwise returns `SHC_FAIL`.

There are two options needs to be provided while calling the method: IP address or domain name of the server and the timeout value in seconds. The most common usage of the method is shown below;

```
if (shApi.shserverisready(targetHost, 5) != shApi.SHC_OK) {
   feedback("! server is not ready yet, should be tried later\n");
   return;
};
```

2.3. Callbacks

Following callbacks are defined as protected method in the API.

```
void cbRemoteMessage(final String msg)
```

The method will be called automatically when the server sends a message. The message argument will be explained in next chapter (Constants and Variables).

```
void cbProgress(final int p)
```

The method which should be overridden to handle progress callback generated by the `shconnect()` method while communicating with the server. It's single argument is the progress value in percent (in range of 0-100). A ProgressDialog may closed when the value reaches to 100 or when the method `cbContentReady()` triggered.

```
void cbConfigReady()
```

The method which should be overridden to handle interim update while `shconnect()` is running and the content is downloading from the server. This callback will be triggered when the "configuration" is downloaded with all it's relevant media properties and ready to use by the custom application. For example, application may set background, logo and welcome text immediately when this callback is triggered while the content processing is still in progress.

```
void cbContentReady()
```

The method which should be overridden to handle final update of `shconnect()`. It means that the `shconnect()` is finished running, all data-types and relevant classes are created and they are ready to use by the custom application. When the method is triggered, user application can start to use all data-structures and classes related with content data.

```
void cbConnectFailed(final int reason)
```

The method which should be overridden to handle errors while `shconnect()` method is running. Once triggered, `shconnect()` stops running and there will no further callbacks be triggered. Error code will be set into calling argument accordingly.

3. Constants and Variables

There are `Integer` constants for error codes (resulting value of some of API functions), option codes while calling some of API functions and `String` constants to use with call-back functions.

3.1. Error Codes

Every functions implemented on native interface will return with an integer value explained below;

```
int SHC_OK   = 0;    // calling function successful
int SHC_FAIL = -1;   // calling function get failed
```

The method `shApiMain.shconnect()` may return following values in addition to above codes and `shApiMain.shcbconnectfailed()` callback method will be triggered accordingly.

```
int SHC_UNSUPPORTED_PLATFORM = 1; // device or platform is not supported
int SHC_L3_DENIED            = 2; // access through routed networks not allowed
int SHC_DEVICE_DISABLED      = 3; // device status is disabled on server side
```

3.2. Calling Arguments

There are only two options used as an argument while calling methods in this API , which are;

```
int SHC_OPT_DOCACHE    = 1; // Store remote content as cache
int SHC_OPT_DONOTCACHE = 2; // Disable caching
```

API will cache media objects such as JPEG or PNG files into local storage in advance to access them quickly whenever it possible. Since the cached objects live in local file system, API will check whether required object is already cached or not, if so, it will be used from the local quickly, otherwise object will be downloaded from the server.

If caching is disabled, media files will not be cached and they will be downloaded from server every time.

The cache folder will be determined by the custom application. This folder can be removed freely before the API calls when necessary.

3.3. Remote Messages

When server sends a message, API triggers a callback method with `String` message argument.

Following message keywords will be sent by the server upon admin request for this particular endpoint;

- `import` Update Message (Banner, header or welcome text)
- `commit` Restart the Application (upon admin request for this endpoint)
- `reboot` Reboot the system
- `clean` Update the firmware (custom application will update itself, cleans the cache etc.)

- `kill` Stop running the application
- `emerg` Emergency mode (*)
- `popup` Pop-Up message received for this STB.
- `insist` Bulk Pop-Up message received for all STBs in the system (*)
- `stbscr` Scrolling text message received for this STB (*)
- `txtscr` Bulk Scrolling Text message received for all STBs in the system (*)

Following message keywords will be sent by the server according to operational changes on server side;

- `kill` Suspend (“commit" launched on server side, app may stop running or do nothing)
- `resume` Resume (“commit" finished on server side, reload content, refresh UI, continue running)

This message keyword will be triggered automatically when the endpoint connected and gathered the configuration from the server;

- `import` Update Message (Banner, header or welcome text)

(*) Additional action required to collect information regarding remote message sent by the server. Custom application will call `shApiMain,shgetremotedata(msg_keyword)` method to get relevant data of the message related `msg_keyword`.

Above `String` constants are defined as static variables also in the API like below;

```
String SHC_MESSAGE_BANNERTEXT = "import";
String SHC_MESSAGE_RESTART = "commit";
String SHC_MESSAGE_SUSPEND = "kill";
String SHC_MESSAGE_RESUME = "resume";
String SHC_MESSAGE_REBOOTSYSTEM = "reboot";
String SHC_MESSAGE_FWUPDATE = “clean”;
String SHC_MESSAGE_EMERGENCY = "emerg";
String SHC_MESSAGE_POPUP = "popup";
String SHC_MESSAGE_BULKPOPUP = "insmsg";
String SHC_MESSAGE_SCROLL = "stbscr";
String SHC_MESSAGE_BULKSCROLL = "txtscr";
```

Custom application may triggered if emergency mode (`SHC_MESSAGE_EMERGENCY`) is active, Pop-Up message (`SHC_MESSAGE_POPUP`) exists or Scrolling Text (`SHC_MESSAGE_SCROLL`) exists after the method `shApi.shconnect()` returns `SHC_OK`. If so, custom application will perform relevant action accordingly. For example, if emergency mode is active, GUI will not shown but the emergency message shown (and the app stop doing normal activities in this mode) until the emergency mode triggered again and empty `String` returns with `shApiMain.shgetremotedata(SHC_MESSAGE_EMERGENCY)` call.

When the method `shApiMain,shgetremotedata()` called against `SHC_MESSAGE_SCROLL` and `SHC_MESSAGE_BULKSCROLL` a list of variables splitter with char(27) returned in the single `String` element respectively:

`Text Size`	An integer which represents the pixel size of the text
`Text Color`	Color with alpha channel in HTML notation like `#FFFFFFFF` for white
`Text Back Color`	Color with alpha channel in HTML notation like `#FFAABBCC` for black
`Number of Text`	The total number of appearance of the text
`Number of Loop`	The total number pass of the group of texts (20 means infinite).
`Scrolling Speed`	Speed can be set according to following values;

24	Slowest
12	Slower
6	Slow
3	Normal

```
2      Fast
1      Faster
0.7    Fastest
```

`Bottom Margin`	Margin from bottom in pixels.
`Text`	Text to be displayed.

3.4. Variables

There are a few variables associated with data fetched from the server:

```
Hashtable<String, String> tableconfig = null;
ContentList contentlist = null;
WeatherItem weatherlist = null;
```

The configuration fetched from the server will be converted to an `hashtable` called "`tableconfig`", content will be converted to a `ContentList` class called "`contentlist`" and weather information will be stored as `WeatherItem` class called "`weatherlist`".

There is also required methods implemented to access such data from the object itself.

4. Configuration

Kylone Endpoint API provides a statically defined `hashtable` object called `tableconfig` to hold configuration data which set on server side. Custom application may use such values to maintain compatibility with server.

The value of this variable set as `null` by default. The `hashtable` object will be created and assigned to `tableconfig` variable immediately when configuration-data is arrived. Custom application will have a callback when the configuration-data is arrived and it will use this variable to access whole configuration data.

Following configuration properties are defined on server side and values can be accessible using "key" name with `getConfigVal()` static method. Using such values to make UI configurable through Kylone server is totally up to custom applications.

Key	Sample Value	Description

`i18n`	`en`	UI Language. This can be one of the following values;
		
		`"en", English`
		`"de", German`
		`"fr", French`
		`"ru", Russian`
		`"zh", Chinese`
		`"th", Thai`
		`"tr", Turkish`
		`"ar", Arabic`
		`"el", Greek`

`logo`	`[resource]`	Main logo (PNG image) resource (URL).
`bgnd`	`[resource]`	Main background (PNG or JPEG image) resource (URL)
`opac`	`0`	Main background transparency in percent (0-100).
`splash`	`[resource]`	Splash or Boot picture in PNG format.

strm	kylonev2	Streaming engine (currently it is Kylone v2).
surl	m3u	Streaming engine access model. This can be one of the following values; "m3u", Resources are configured to use m3u playlists "raw", Resources are configured to use RAW HTTP socket
mmtype	csr	Main Menu Type or Theme. This can be one of following values; "def", Classic "pre", Facility (most used by hospitality organisations) "csr", Consumer Style Custom application may have a few different types of UI design (Theme) and may display them according to selection above which configured on server side.
xfsfast	yes	Effects/Animations on UI can be configured as enabled/disabled on server side: "yes", Disable effects (fast UI, for all hardwares) "no" , Enable effects (strong hardware may needed) Custom application may obey on this value and enable/disable its UI effects accordingly to be compatible with infrastructure (i.e. only for hardware utilised as STB).
mmshdcl	black	Shadow (or shade) colour. Generally used to suppress whatever is on the screen and putting something on top of it.
mmstyle	rec	Button style of main menu items. This can be one of following values; "rec", Menu buttons created using vector drawings. “img", Menu buttons created using images.
mmitatc	orange	Text colour of active (selected or highlighted) menu item.
mmitasc	black	Style colour of text on active menu item.
mitxtcl	white	Default text colour of menu item.
mitxtsc	silver	Default style colour of text of menu items.
mmitidl	[resource]	Background image (URL) of passive (default) menu item.
mmitact	[resource]	Background image (URL) of active menu item.
langbut	[resource]	Background image (URL) of language selection button.
appclose	[resource]	Background image (URL) of Close/Home/Back button.
appcatico	[resource]	Indicator image (URL). It can be used to indicated selected category or playing item etc.
appbusypng	[resource]	Busy indicator image (URL).
mcbnostar	[resource]	An “empty” “star” image (URL) to be used to show rating.
mcbstar	[resource]	An “filled” “star” image (URL) to be used to show rating.
mcbspk0	[resource]	An image (URL) to be used to show volume level as 0%.
mcbspk1	[resource]	An image (URL) to be used to show volume level as 33%
mcbspk2	[resource]	An image (URL) to be used to show volume level as 66%

mcbspk3	[resource]	An image (URL) to be used to show volume level as 100%
mcbaspect	[resource]	An image (URL) to be used to put “aspect ratio” button.
mcbasprhv	[resource]	An image (URL) to be used to put “cover-all-screen” button.
mcbdock	[resource]	An image (URL) to be used to put “dock” button.
mcbundock	[resource]	An image (URL) to be used to put “undock” button.
mcbgrip	[resource]	An image (URL) to be used to put “grip” button.
mcbmove	[resource]	An image (URL) to be used to put “move” button.
mcbplay	[resource]	An image (URL) to be used to put “play” button.
mcbpause	[resource]	An image (URL) to be used to put “pause” button.
mcbstop	[resource]	An image (URL) to be used to put “stop” button.
mcbprev	[resource]	An image (URL) to be used to put “prev” button.
mcbnext	[resource]	An image (URL) to be used to put “next” button.
mcbback	[resource]	An image (URL) to be used to put “back” button.
mcbbufb	8	Minimum buffer value for manual buffering. This value is dependent on media-player backend of hardware utilised. The media-player may use this value as minimum value to play stream.
mcbbufm	16	Medium buffer value for manual buffering. The media-player may use this value as medium value to start buffering again.
mcbbuft	24	Maximum buffer value for manual buffering. The media-player may use this value as maximum value to stop buffering.
mcbbufa	1	Auto Buffering (ignore manual buffering values above). “0", Disable auto-buffering (use manual buffering) “1", Enable auto-buffering (ignore manual buffering)
mcbavse	1	Audio/Video time synchronisation while playing streams. “0”, Auto (media-player will handle everting) “1”, Enabled (custom application will manage everything)
mcbavss	1	Stream start-playing method. “1”, Auto “2”, Show still image and wait time-sync “3”, Show blank screen and wait time-sync
mcbavsm	2	Audio/Video time synchronisation handling (customisation requirements): “1”, Use System Default value (on advanced settings) “2”, Use Channel Configuration (override sys-default) “3”, Always use “Video PID” as time source “4”, Always use “Audio PID” as time source “5”, Always use “PCR PID” as time source

mcbavco	1	Channel Override for Audio/Video time synchronisation “1”, No “2”, Use System Default “3”, Use “Video PID” as time source “4”, Use “Audio PID” as time source “5”, Use “PCR PID” as time source
mcbavpo	1	PCR Override for Audio/Video time synchronisation “1”, No “2”, Use “Video PID” as sync source “3”, Use “Audio PID” as sync source
mcbavst	65	Audio/Video time synchronisation threshold (milliseconds).
mcbchsd	350	Channel switch delay (zap) in milliseconds.
mcbtout	10	Timeout while playing: Connection timeout while playing live stream (seconds). Since live-streams may goes offline time by time, custom application may stop playing after waiting given amount of time.
mcbtrcv	7	Recovery: Connection timeout while starting to play live stream (seconds). Since live-streams may temporarily offline, custom application may try to play same stream after given value again and again.
mcbdoct	20	Undocking timeout (seconds).
mcbmenu	3000	OSD timeout (milliseconds).
mcbrnum		Sort and renumber channels (starting from 1) in every category alphabetically according to their names. “true”, sort and renumber empty of “false”, don’t take any action.
mcbsort	true	Sort and renumber channels (starting from 1) alphabetically according to their names (covering all categories). “true”, sort and renumber empty of “false”, don’t take any action.
mcstart	1	Starts playing given live-tv channel (with it's number) immediately when the custom application starts running.
mcpltsd	2	TS Demuxing method “1”, software “2”, hardware (accelerated)
mcpmpfs	2	Fast switch (one of live stream resource to another). “1”, enabled: don't waste so much time to analyse input stream, start as soon as possible. “2”, disabled: analyse input stream correctly and start playing.
mcphtor	3	Number of HTTP Open retry before give up (Live stream).
mcphtrr	2	Number of HTTP Read retry before give up (Live stream).

mcpxato	300	Live TS analyse timeout (milliseconds).
mcbbuf0	[resource]	An image (URL) to be used to show buffer status as “empty” when starting to play stream. It may used when the live-stream is not connected yet also.
mcbbuf1	[resource]	An image (URL) to be used to show connection status as “connected" when starting to play stream.
mcbbuf2	[resource]	An image (URL) to be used to show buffer status as “buffering” when starting to play stream.
mcbbuf3	[resource]	An image (URL) to be used to show buffer status as “buffering” when stream stops playing (when buffer gets empty while playing).
mcbbuf4	[resource]	not used
mcbbuf5	[resource]	An image (URL) to be used to show buffer status as “full” enough. It may used also when the live-stream starts playing.
txtncol	white	General text colour to be used on UI.
txtscol	gray	General text style colour to be used on UI.
txtszm	40	General text size to be used on UI.
txtszt	36	General text size for “titles” to be used on UI.
txtszi	45	General text size for “item-titles” to be used on UI.
navblen	6	Border size (in pixels) of Navigation cursor.
navbcol	orange	Border colour of Navigation cursor.
navatcl	black	Text colour to be used with Navigation cursor.
navatst	silver	Text style colour to be used with Navigation cursor.
navstyle	rec	Navigation cursor style “rec”, rectangles (vector drawing) “img”, pixmaps or images
catvnbgc	transparent	Background colour of a “Category Item”
catvnopa	0.50	Background colour transparency of a “Category Item” (float, 0-1)
catvnrad	5	Border radius of “Category Item"
catvngr2	black	Background gradient-start colour of "Category Item"
catvngr1	purple	Background gradient-stop colour of "Category Item"
catvnbor	gray	Border colour of “Category Item"
catvnact	white	Border colour of active or highlited “Category Item"
catvntcl	white	Text colour of “Category Item"
catvntsc	gray	Text style-colour of “Category Item"
prename		Header text to display on top of the screen - for “Facility" theme
premimg	[resource]	Picture (URL) to display on main screen for “Facility" theme
prelogo	[resource]	Logo image (URL) to display as logo for “Facility" theme
prehdbg	[resource]	Mid-header background image (URL) for “Facility" theme

pretbgi	[resource]	Top-header background image (URL) for “Facility" theme
prewfic	[resource]	WiFi icon image (URL)
prewftx		WiFi description text (SSID, password etc) for “Facility" theme
csrmfc	#c53329	Navigation cursor border colour of “C-Style” theme
csrmfa	#ff4c3e	Navigation cursor border colour for active or highlighted items of “C-Style” theme
csrsbt	[resource]	Settings button image (URL) for “C-Style" theme
csrlogo	[resource]	Logo image (URL) to display as logo for “C-Style” theme
csrback	[resource]	Background image (URL) for “C-Style" theme.

5. Content

Kylone Endpoint API will provide it’s own class-type definitions for processing structured data sent by the server. In order to understand such data-types and it’s related classes well, we will have a look into data coming from the server first.

Kylone server will always send data in XML format. After doing request to server, resulting XML data will be converted to a structured data using such classes which are implemented in this API to access and use content data easily with other UI components.

```
<content id="item">...</content>
<content id="prefs">...</content>
<content id="movie">...</content>
<content id="music">...</content>
<content id="clip">...</content>
<content id="tv">...</content>
<content id="radio">...</content>
...
<content id="locs">...</content>
```

In the basic form of XML document seen above, there are `content` sections which are defining content-groups with their `id` property.

```
<content id=“music" rank="">
   <category id="28">
      <item cat="28">...</item>
      <item cat=“28">...</item>
      ...
   </category>
   <category id=“21" rank="">
      <item cat="21">...</item>
      <item cat=“21">...</item>
      ...
   </category>
   <category id=“18" rank="">
      <item cat="18">...</item>
      <item cat="18">...</item>
      ...
   </category>
   ...
</content>
```

```
...
```

Each `content` section has it's own `category` list and each `category` has an `id` which holds the category value and it's own `item` list inside.

The category value defined as `id` is a custom information set by the server admin and they are `Integer` values.

Each `item` section holds a list of values which we will call them as `attributes`. Following is an example about attributes of an `item` which belongs to "`music`" content group;

```
<item>
   <attr id="src">record name of song</attr>
   <attr id="txt">display name of the song</attr>
   <attr id="title">track name of the song</attr>
   <attr id="artist">artist of band name</attr>
   <attr id="album">album name</attr>
   <attr id="year">year value</attr>
   <attr id="type">genre of the song</attr>
   <attr id="track">track number of the song in album</attr>
   <attr id="cat">18</attr>
</item>
```

Each content-group may have items inside it with different attribute `id` values according to their functions. For example a movie item may have description or a poster image but a music item will have a cover art image etc.

Custom applications may directly use such resulting XML data instead of using ready to use classes when necessary.

5.1. Attributes

Attributes are represented as **`ContentAttribute`** class in the API. It holds the list of attributes with key/value (`String/String`) pair and it has required methods implemented to access such attributes.

```
<attr id="title">track name of the song</attr>
<attr id="artist">artist or band name</attr>
<attr id="album">album name</attr>
```

`int size()`

Returns number of attributes stored in the object. According to above data-set this will return `3`.

`Enumeration<String> keys()`

Returns an Enumeration object to get a list of keys of attributes in the object.

```
for (Enumeration<String> e = ContentAttribute.keys(); e.hasMoreElements();)
   String key_name = e.nextElement();
```

Above example will return key names as "`title`", "`artist`", "`album`" on each call for the data-set shown above.

`String[] getList()`

Returns keys as a string list. This will return a `String[]` object with following values according to data-set shown above.

```
{“title”, “artist”, “album”}
```

`String get(String key)`

Returns the value of attribute with given name.

```
ContentAttribute.get(“title”); // returns “track name of the song"
```

`String get(int idx)`

Returns the value of attribute with given index.

```
ContentAttribute.get(1); // returns “artist or band name"
```

This function will be used in conjunction with `String[]` returned by the `getList()` method and the index value will be same with the index value of returning `String[]` object of `getList()` method. For example, custom application may use the `getList()` output in a ListView and then it may get particular value of an attribute by using the index of the list-item.

5.2. Items

Items are represented as **`ContentItem`** class in the API. It holds the list of items with key/value (`String/ContentAttribute`) pair and it has required methods implemented to access such items and it’s attributes.

```
<item>
   <attr id=“txt”>item 0</attr>
    ...
</item>
<item>
   <attr id=“txt”>item 1</attr>
   <attr id=“title”>title value of item 1</attr>
    ...
</item>
<item>
   <attr id=“txt”>item 2</attr>
    ...
</item>
<item>
   <attr id=“txt”>item 3</attr>
    ...
</item>
```

The key values of the object determined by it’s `“txt”` attribute. And the value of Item will be a `ContentAttribute` object which holds it’s attributes.

`int size()`

Returns number of items stored in the object. According to above data-set this will return `4`.

`Enumeration<String> keys()`

Returns an Enumeration object to get a list of keys of items in the object.

```
for (Enumeration<String> e = ContentItem.keys(); e.hasMoreElements();)
   String key_name = e.nextElement();
```

Above example will return key names as “item 0”, “item 1”, “item 2”, “item 3” on each call for the data-set shown above.

`String[] getList()`

Returns keys as a string list. This will return a `String[]` object with following values according to data-set shown above.

```
{“item 0”, “item 1”, “item 2”, “item 3”}
```

`ContentAttribute get(String key)`

Returns the value as `ContentAttribute` object of the item with given name.

```
ContentItem.get(“title 1”); // returns a ContentAttribute object
```

`ContentAttribute get(int idx)`

Returns the value as `ContentAttribute` object of the item with given index.

```
ContentItem.get(1); // returns a ContentAttribute object
```

This function will be used in conjunction with `String[]` returned by the `getList()` method and the index value will be same with the index value of returning `String[]` object of `getList()` method.

`String get(String itemkey, String attributekey)`

Returns the attribute value according to given item and attribute keys if there is or `null` if no such item or attribute exists. With this method, it is possible direct access to particular attribute of an item in the list.

```
ContentItem.get(“item 1”, “title”); // returns “title value of item 1”
```

`String get(int itemindex, String attributekey)`

Returns the attribute value according to given item-index and attribute key if there is or `null` if no such item or attribute exists.

```
ContentItem.get(1, “title”); // returns “title value of item 1”
```

5.3. Categories

Categories are represented as **`ContentCategory`** class in the API. It holds the list of categories with key/value (`String/ContentItem`) pair and it has required methods implemented to access such categories and its items.

```
<category id="28" rank="">...</category>
<category id="21" rank="">...</category>
   <item>...</item>
   <item>...</item>
    ...
```

```
<category id="18" rank="">...</category>
```

`int size()`

Returns number of category stored in the object. According to above data-set this will return `3`.

`Enumeration<String> keys()`

Returns an Enumeration object to get a list of keys of categories in the object.

```
for (Enumeration<String> e = ContentCategory.keys(); e.hasMoreElements();)
   String key_name = e.nextElement();
```

Above example will return key names as `“28”, “21”, “18”` on each call for the data-set shown above.

`String[] getList()`

Returns keys as a string list. This will return a `String[]` object with following values according to data-set shown above.

```
{“28”, “21”, “18”}
```

`ContentItem get(String key)`

Returns the value as `ContentItem` object of the category with given name.

```
ContentCategory.get(“21”); // returns a ContentItem object
```

`ContentItem get(int idx)`

Returns the value as `ContentItem` object of the category with given index.

```
ContentCategory.get(1); // returns a ContentItem object
```

This function will be used in conjunction with `String[]` returned by the `getList()` method and the index value will be same with the index value of returning `String[]` object of `getList()` method.

`ContentAttribute get(String categorykey, String itemkey)`

Returns the value as `ContentAttribute` object according to given category key and item key if there is or `null` if no such category or item exists. With this method, it is possible direct access to attribute object of particular item of a category in the list.

```
ContentCategory.get(“21”, “item 1”); // returns a ContentAttribute object
```

`ContentAttribute get(String categorykey, int itemindex)`

Returns the value as `ContentAttribute` object according to given category key and item-index if there is or `null` if no such category or item exists. With this method, it is possible direct access to attribute object of particular item of a category in the list.

```
ContentCategory.get(“21”, 1); // returns a ContentAttribute object
```

`String get(String categorykey, String itemkey, String attributekey)`

Returns the value of attribute according to given category key, item key and attribute key if there is or `null` if no such category, item or attribute exists.

```
ContentCategory.get("21", "item 1", "title"); // returns "title value of item 1"
```

`String get(String categorykey, int itemindex, String attributekey)`

Returns the value of attribute according to given category key, item-index and attribute key if there is or `null` if no such category, item or attribute exists.

```
ContentCategory.get("21", 1, "title"); // returns "title value of item 1"
```

`String getRank()`

Returns the value to use for ranking. Rank value may be an empty string otherwise it holds a value which should be interpreted as integer. Custom application may do sorting using this value from lower to greater in lists.

5.4. Content Groups

Content-Groups are represented as **`ContentList`** class in the API. It holds the list of content groups with key/value (`String/ContentCategory`) pair and it has required methods implemented to access such categories and it's items.

```
<content id="movie">...</content>
<content id="music">...</content>
   <category id="28" rank="">...</category>
   <category id="21" rank="">...</category>
   ...
<content id="tv">...</content>
<content id="radio">...</content>
```

The key values of the object determined by it's `"id"` property. And the value will be a `ContentCategory` object which holds it's items.

This data-type may called as "*rootitems*" or "*menuitems*" also. User application may display list of items in this object as "Main Menu Items" on the UI. Please check next chapter for more details about menu-items.

The API has a `ContentList` type static variable called `"contentlist";`

```
public static ContentList contentlist = null;
```

The value of this variable set as `null` by default. The `ContentList` object will be created and assigned to this variable immediately when content-data is arrived. Custom application will have a callback trigger when the content-data is arrived and it will use this variable to access whole content from single object.

`int size()`

Returns number of group stored in the object. According to above data-set this will return `4`.

`Enumeration<String> keys()`

Returns an Enumeration object to get a list of keys of content-groups in the object.

```
for (Enumeration<String> e = ContentList.keys(); e.hasMoreElements();)
   String key_name = e.nextElement();
```

Above example will return key names as “movie”, “music”, “tv”, “radio” on each call for the data-set shown above.

String[] getList()

Returns keys as a string list. This will return a `String[]` object with following values according to data-set shown above.

```
{“movie”, “music”, “tv”, “radio”}
```

ContentCategory get(String key)

Returns the value as `ContentCategory` object of the content-group with given name.

```
ContentList.get(“music”); // returns a ContentCategory object
```

ContentCategory get(int idx)

Returns the value as `ContentCategory` object of the content-group with given index.

```
ContentList.get(1); // returns a ContentCategory object for “music”
```

This function will be used in conjunction with `String[]` returned by the `getList()` method and the index value will be same with the index value of returning `String[]` object of `getList()` method.

ContentItem get(String contentkey, String categorykey)

Returns the value as `ContentItem` object according to given group and category keys if there is or `null` if no such group or category exists. With this method, it is possible direct access to Item object of particular category of a group in the list.

```
ContentList.get(“music”, “21”); // returns a ContentItem object
```

ContentAttribute get(String contentkey, String categorykey, String itemkey)

Returns the value as `ContentAttribute` object according to given group, category and item keys if there is or `null` if no such group or category or item exists.

```
ContentList.get(“music”, “21”, “item 1”); // returns a ContentAttribute
```

ContentAttribute get(String contentkey, String categorykey, int itemindex)

Returns the value as `ContentAttribute` object according to given group key, category key and item-index if there is or `null` if no such group or category or item exists.

```
ContentList.get(“music”, “21”, 1); // returns a ContentAttribute object
```

String get(String contentkey, String categorykey, String itemkey, String attributekey)

Returns the value of attribute according to given group, category, item and attribute keys if there is or `null` if no such group, category, item or attribute exists.

```
ContentList.get(“music”, “21”, “item 1”, “title”);
```

`String get(String contentkey, String categorykey, int itemindex, String attributekey)`

Returns the value of attribute in same way but using item-index if there is or `null` if no such group, category, item or attribute exists.

```
ContentList.get(“music”, “21”, 1, “title”);
```

6. Using the Content

Attribute keys for different items in different content-groups will have different set of definitions. Possible content-groups and its usage will be explained in next chapters.

6.1. Menu Items

There is content-group called `"item"` holds menu-items. Their `“src”` attribute is same with the content-group IDs in general.

```
<content id="item">
   <category id="0" rank="">
      <item>...</item>
      <item>
         <attr id="src">music</attr>
         <attr id="txt">Music</attr>
         <attr id="ena">1</attr>
         <attr id="ord">3</attr>
         <attr id="logo">...</attr>
         <attr id="bgnd">...</attr>
         <attr id="opac">60</attr>
         <attr id="sub">0</attr>
      </item>
      <item>...</item>
      <item>...</item>
   </category>
</content>
```

As shown above example, there is category with ID zero (`"0"`) even if there is no need to use categories to maintain compatibility with our organised data structure. The category with ID zero (`"0"`) may used to represent all items in particular content-group also.

In order to find menu-items to show on UI of the custom application, items under category(`"0"`) of content-group(`"item"`) should be used. When a particular menu-item is selected, its `“src”` attribute will be used to access content-groups to find related content shown below.

```
<content id=“music">...</content>
   <category id="28" rank="">...</category>
   <category id="21" rank="">...</category>
   ...
</content>
```

It is possible to get list of menu-items using static variable like below;

```
shApiMain.ContentItem menuitems = shApiMain.contentlist.get("item", “0");
```

Following example can be used to get a list of "*rootitems*" and to print name of each;

```
feedback("\n> showing list of root items in content as an example\n");
String[] rootlist = m_api.contentlist.getList();
for (int i = 0; i < m_api.contentlist.size(); i++) {
   feedback("   - " + rootlist[i] + "\n");
}
feedback("\n");
```

Above example will produce following output;

```
> showing list of root items in content as an example
   - movie
   - radio
   - tv
   - prefs
   - clip
   - item
   - locs
   - music
```

Following example can be used to get list of “Main Menu Items” configured on server side and to print name of each;

```
shApi.ContentAttribute attr = shApi.contentlist.get("item", "0", "music");
String itemtext = attr.get("txt");

feedback("  - item: " + itemtext + "\n");
String list[] = attr.getList();
for (int i = 0; i < list.length; i++) {
   feedback("      " + list[i] + ": " + attr.get(i) + "\n");
}
```

Above example will produce following output;

```
> showing list of Menu Items as an example
   - Video Clip
   - Music
   - Settings
   - Television
   - Movie
   - Weather
   - Information
   - Radio
```

Another example shown below will print all attributes of “music” which is from “Main Menu Item” list.

```
feedback("> showing list of Menu Items as an example\n");
String[] list = m_api.contentlist.get("item", "0");
for (int i = 0; i < list.length; i++) {
   feedback("   - " + list[i] + "\n");
}
feedback("\n");
```

Above example will produce following output;

```
- item: Music
    cat: 0
    txt: Music
    ord: 3
    ena: 1
    ptxt:
    sub: 0
    opac: 60
    bgnd: ...
    logo: ...
    src: music
```

Key	Sample Value	Description
`src`	`music`	Attribute that match to content-group ID.
`txt`	`Music`	Display name of the item.

`ena`	`1`	Attribute that holds status of this item
		
		`"0", Disabled, item will not functioning`
		`"1", Enabled`

`ord`	`3`	Sequence number of the item. Items may sorted using this attribute while placing to UI.
`logo`	`[resource]`	An image (URL) to be used as button or background image of menu item.
`bgnd`	`[resource]`	An image (URL) to be used as background image of the new page when an item is selected. It may used for any other purpose also.
`opac`	`60`	Transparency of the background image (percent, 0-100).

`sub`	`0`	Attribute that holds sub-menu flag of this item. If the item is sub-menu, then it a new set of menu will be displayed on UI using the items's `src` attribute.
		
		`"0", there is no sub-menu for this item.`
		`"1", there is sub-menu defined for this item.`

`ptxt`		A text to be used as description of the item. It may used for any other purpose also.
`cat`	`0`	Category ID of this item (copied from the parent category id).

6.2. Sub Menu Items

Sub-menus are defined in same way with the main menu items. The difference is that the sub-menus are located into a category with id called sub-menu name.

Please check following example.

```
<content id="item">
   <category id="0" rank="">
      <item>...</item>
      <item>
         <attr id="src">info</attr>
         <attr id="sub">1</attr>
      </item>
      <item>...</item>
   </category>
   <category id="info" rank="">
         <item>...</item>
         <item>...</item>
         ...
  </category>
</content>
```

In above example, since there is “sub” flag set as “1”, the “info” menu item is defined as sub-menu. In this situation, the custom application should look into another category with id “info” which is the name of the item too. So, items of sub-menu will be used from that category.

Please keep in mind that there may nested sub-menu definitions.

6.3. Preferences

This content group called as “prefs” and holds the preferences menu of the application.

```
<content id="prefs">
```

Following example will get all items this content-group.

```
shApiMain.ContentItem items = shApiMain.contentlist.get(“prefs”, “0”);
```

Key	Sample Value	Description
src	language	not used
txt	Language	Display name of the item.
ena	1	Attribute that holds status of this item “0”, Disabled, item will not functioning “1”, Enabled
logo	[resource]	An image (URL) to be used as button or background image of the item.
cat	0	Category ID of this item (copied from the parent category id).

6.4. Movie

This content group called as “movie” and holds the Video On Demand content.

```
<content id="movie">
```

Following example will get all items this content-group.

```
shApiMain.ContentItem items = shApiMain.contentlist.get(“movie”, “0”);
```

Key	Sample Value	Description
src	[resource]	URL to be used to play movie
txt		Title or display name of the movie
itxt		A brief description of the movie
idir		A text to be used to display “directors” of the movie
star		A text to be used to display “cast” of the movie
irate	7.4	A float number (0.0-10.0) to be used to display “rating” of the movie.
kind	Advanture	A text to be used to display “kind” of the movie
dur	2h 22min	A text to be used to display “duration” of the movie
year	2018	A text to be used to display “year” of the movie
logo	[resource]	An image (URL) to be used as poster image of the movie.
cat	0	Category ID of this item (copied from the parent category id).

6.5. Music

This content group called as “music” and holds the Video On Demand content.

```
<content id="music">
```

There will be no category with id “0” for “music”. Following example will get all categories this content-group.

```
shApiMain.ContentCategory cats = shApiMain.contentlist.get(“music”);
```

Key	Sample Value	Description
src	[resource]	URL to be used to play song
fbp		File base path of the song on the server side.
txt		Title or display name of the movie
title		A text to be used to display “name” of the song
artist		A text to be used to display “singer” or “band" of the song
album		A text to be used to display “album” of the song

year	2018	A text to be used to display “year” of the album or song
type	20	not used
track	4/15	A text to be used to display “track” information in album of the song
cat	0	Category ID of this item (copied from the parent category id).

Most of the attributes of the music item will be determined from the music file itself using ID3 tags including cover-arts. So, it is important to have music files with proper ID3 tags and cover-art resource.

Cover-art can be fetched from the server using an API call with file-base-path option. Following is the method definition in the API;

```
Drawable getCoverArt(final String hostaddr, final String mediabasepath)
```

Following is the usage of such method to fetch cover-art from the server;

```
Drawable res = shApi.getCoverArt(targetHost, itemattributes.get(“fbp"));
```

6.6. Music Video

This content group called as “clip” and holds the Music Video content.

```
<content id="clip">
```

Following example will get all items in this content-group.

```
shApiMain.ContentItem items = shApiMain.contentlist.get(“clip”, “0”);
```

Key	Sample Value	Description
src	[resource]	URL to be used to play music video.
txt		Title or display name of the music video.
star		A text to be used to display “singer” or “band" of the music video.
logo	[resource]	An image (URL) to be used as cover-art of the music video.
cat	0	Category ID of this item (copied from the parent category id).

6.7. Live TV

This content group called as “tv” and holds the Live TV content.

```
<content id="tv">
```

Following example will get all items in this content-group.

```
shApiMain.ContentItem items = shApiMain.contentlist.get(“tv”, “0”);
```


Key	Sample Value	Description
src		not used
txt		Title or display name of the live tv channel.
syn		Attribute that holds A/V sync source of the live tv channel. “sys”, System default (use configuration parameter) “aid”, force to use Audio PID “vid”, force to use Video PID “pcr”, force to use PCR PID
logo	[resource]	An image (URL) to be used as logo of the live tv channel.
rrc	[resource]	URL to be used to play live tv channel for STBs mostly based on Linux/ffmpeg.
arc	[resource]	URL to be used to play live tv channel for Android based systems.
xrc	[resource]	URL to be used to play live tv channel for iOS based systems.
raw	[resource]	URL to be used to play live tv channel using RAW HTTP socket.
rpc	[resource]	URL to be used to play live tv channel for Desktop PCs mostly based on ffmpeg.
xpc	[resource]	URL to be used to play live tv channel for other Desktop PCs (playlist based).
chn	1	Sequence number of the live tv channel.
cat	0	Category ID of this item (copied from the parent category id).

6.8. Live Radio

This content group called as “radio” and holds the Live Radio content.

```
<content id="radio">
```

Following example will get all items in this content-group.

```
shApiMain.ContentItem items = shApiMain.contentlist.get(“radio”, “0”);
```

Attributes and usage of Live Radio content group is same with the Live TV.

7. Helper Methods

`int getAlpha(final String val)`

Static method for translating transparency to alpha value. On server side, only transparency value will be given in range of 0-100. So, it needs to be adopted to alpha (0-255) like opacity. This method will return alpha value using server side transparency accordingly.

`Drawable createBitmapFromURL(final String url)`

Static method for creating a `Drawable` object form given URL. The URL may a cached local file or may a point to a resource on server side. In either way, the URL will be read and `Drawable` object will be created accordingly.

`Drawable getCoverArt(final String hostaddr, final String mediabasepath)`

Cover arts are generally stored in media files with ID3 tags. There is a server-side implementation to get such data from the media file remotely. This method fetch media cover-art form the server using base path of the media file. Custom application may extract such info from the music file with it's own implementation too.

`Drawable createBitmapFromURLScaled(final String url, int width)`

Static method for creating a `Drawable` object form given URL and scales it while keeping the aspect-ratio of the media.

`void killMyself(int reload)`

This method is a callback and it may called from the native implementation when the server requests command to re-launch or terminate the application.

`String inetConnected(int typ)`

This method is implemented for custom application to check whether internet connectivity exists or not. It may used before trying to connect to server.

`String wifiConnected(int typ)`

This method is implemented for custom application to check whether wifi connectivity exists or not. It may used before trying to connect to server.

`String gethwaddr(final String iface)`

This method implemented to use directly with native interface. The MAC address (or resulting unique ID) will be used to track device on control panel of the server.

`String getdevinfo()`

This method implemented to use directly with native interface. The device information will be used to give additional information on control panel of the server.

`String getsysname()`

This method implemented to use directly with native interface. The system name information will be used to give additional information on control panel of the server.

`String getlocale()`

This method implemented to use directly with native interface. The locale information will be used to give additional information on control panel of the server.

Custom application may use this information to determine UI language too.

8. Examples

8.1. Extending the API Class

Following example will show how to extend API class to use with custom application by overriding some required callback functions.

Since such methods may change the UI, it should be run in UI-thread. So, such required tasks also implemented in this example utilising `runOnUiThread()` or `postDelayed()` methods.

You can find a complete running example application from the API distribution package.

```
// Extend shApiMain to override some required functions (callbacks)
private class shApi extends shApiMain {

   /*
      It is mandatory to create constructor and save context
      which will be used in static functions in API class.
   */
   public shApi(Context ctx) {
      m_apiContext = ctx;
      Log.v("shApi", "constructed");
   }

   /*
      RemoteMessage callback may triggered as per config changes on server side.
      Server will send such info as "message" to inform app.
   */
   @Override
   protected void cbRemoteMessage(final String msg) {
      m_handler.postDelayed(new Runnable() {
         @Override
         public void run() {
            feedback("~ cbRemoteMessage(" + msg + "): ");
         }
      }, 50);
   }

   /*
      Progress callback
      API will call this function while downloading the content. The argument
      is an integer value in range 1-100 which is in percent.
   */
   @Override
   protected void cbProgress(final int p) {
      m_handler.postDelayed(new Runnable() {
         @Override
         public void run() {
           progressdialog.setProgress(p);
           feedback("progress: " + p + "\n");
         }
      }, 50);
   }

   // Handling connection errors
   @Override
   protected void cbConnectFailed(final int reason) {
      runOnUiThread(new Runnable() {
         @Override
         public void run() {
            progressdialog.setProgress(0);
            progressdialog.cancel();
            feedback("! connect failed with reason code: " + reason + "\n");
         }
      });
   }

   // Handling config-ready trigger
   @Override
   protected void cbConfigReady() {
      m_handler.postDelayed(new Runnable() {
         @Override
```

```
        public void run() {
            feedback("~ confguration is ready\n");
            /*
               Since config is arrived, we can utilise some values:
               Main background and logo image can be set at this stage.
               There is also an example usage of createBitmapFromURL() API call below.
            */
            Drawable res = shApi.createBitmapFromURL(shApi.getConfigVal("bgnd"));
            if (res != null) {
               appback.setScaleType(ImageView.ScaleType.CENTER_CROP);
               /*
                  It is possible to use “opac” config-property also.
                  There is an API call for converting such value to “alpha” too.
                */
               appback.setImageAlpha(shApi.getAlpha(shApi.getConfigVal("opac")));
               appback.setImageDrawable(res);
               appback.setVisibility(View.VISIBLE);
               feedback("~ background image set\n");
            }
            res = shApi.createBitmapFromURL(shApi.getConfigVal("logo"));
            if (res != null) {
               logoview.setScaleType(ImageView.ScaleType.CENTER_INSIDE);
               logoview.setImageDrawable(res);
               logoview.setVisibility(View.VISIBLE);
               feedback("~ logo image set\n");
            }
        }
    }, 50);
}

// Handling content-ready trigger
@Override
protected void cbContentReady() {
    m_handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            progressdialog.setProgress(0);
            progressdialog.cancel();
            feedback("~ content is ready\n”);

            /*
               When content arrived, shApiMain.contentlist variable
               (shApiMain.ContentList) class will be constructed.
               You can access all content information through this constructed class

               Following example shows how to get list of main content items.
               It can be determined whether there is Main Menu, Movie, TV, Music etc
               contents exists or not.
            */
            feedback("\n> showing list of root items in content as an example\n");
            String[] rootlist = shApi.contentlist.getList();
            for (int i = 0; i < shApi.contentlist.size(); i++) {
               feedback("   - " + rootlist[i] + "\n");
            }
            feedback(“\n");

            /*
               Below example shows how to get list of Main Menu content (with category all = 0)
               We will print details with calling feedback function and add a Button for every
               Menu Item.
            */
            shApi.ContentItem menuitemsinallcategory = shApi.contentlist.get("item", "0");
            if (menuitemsinallcategory != null) {
               feedbackPrintListItems("showing list of Menu Items as an example",
                   menuitemsinallcategory);
               addMenuButtons(menuitemsinallcategory);
            }

            /*
               Another example below shows to access list of Movies. Instead of using
               category id directly from get() method of shApiContentList, we will walk over
               on the data structure.
            */
            shApi.ContentCategory movies = shApi.contentlist.get("movie");
            if (movies != null) {
               shApi.ContentItem movieinallcategory = movies.get("0");
               if (movieinallcategory != null) {
                  feedback("\n> showing list of Movies with attributes as an example\n");
                  String[] l = movieinallcategory.getList();
```

```
                for (int i = 0; i < movieinallcategory.size(); i++) {
                    shApi.ContentAttribute movieattr = movieinallcategory.get(i);
                    if (movieattr == null)
                        continue;
                    feedbackPrintAttributes(l[i], movieattr);
                    /*
                        We will just display the first item by using break below;
                        Please remove it to see all items with attributes in the movie list
                    */
                    break;
                }
            }
        }
        feedback("\n> click one of the Menu Buttons to show its content in ListView\n");
        setButtonClickListeners();
    }
  }, 50);
 }
}
```

Kylone Technology International Limited
610, 6/F, B.36, Chentian Industrial Park, Baotian 1st RD.
518055, Xixiang Town, Bao'an District, Shenzhen, China
Phone: +86 (755) 26501345
Fax: +86 (755) 26549326
http://kylone.com/ , sales@kylone.com

TBS Technologies International Limited
610, 6/F, B.36, Chentian Industrial Park, Baotian 1st RD.
518055, Xixiang Town, Bao'an District, Shenzhen, China
Phone: +86 (755) 26501345
Fax: +86 (755) 26549326
sales@tbsdtv.com http://www.tbsiptv.com/