Metadata and API Requirement Guide for Content Partners

Version 1.1.2

VIDAA US Inc.

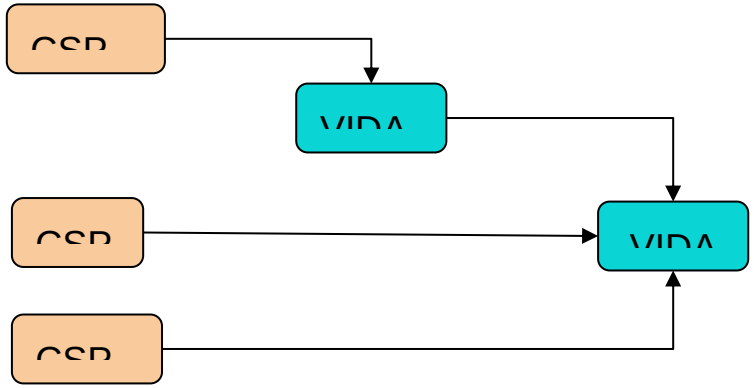
Change Log

Updated On	Version	Brief	Changer	Description
2019-10-15	1.0.1	Update “screenshots” field	Luan Chunhua	remove the mandatory marker of Field "screenshots" in "3.2. Movies, Short Videos, and UGC", "3.3. TV series" and "3.4. Sports"
2019-10-15	1.0.2	3.6. Live Channels	Luan Chunhua	change data model of "3.6. Live Channels" to three tables.
2019-11-20	1.0.3	Section 2 and 3	Bob Lu	3.4 change video_type to show_type 2.6 & 2.7 Newly added
2019-11-21	1.0.4	Updated the field description in Section 3	Bob Lu	Updated the description of streaming_rights, pricing_info, and playback_urls
2019-11-22	1.0.5	Music definition added	Bob Lu	Section 3.7 Added metadata definition for music content
2019-12-06	1.1.0	Add figures, API definition and code sample	Chunhua Luan Bob Lu	1. Move CPS’s confirmation part in checklist file 2. Add integration figures, API definition and code samples
2019-12-09	1.1.1	Small changes on Live_schedule	Chunhua Luan	Added start time and end time for live_schedule
2019-12-17	1.1.2	Add images specifications	Bob Lu	1. Add image specification for Apps 2. Add image size requirement in appendix.

Index

1. Content integration and dispatching structure

We need to integrate all of CSP's metadata information so that our content operation team will have the ability to arrange proper contents for different areas. The data flow is basically like the following figure.



In following sections, we will first describe the overall API requirement in Section 2, then sample API samples with our recommended pattern in Section 3. And all the required metadata fields are described in Section 4.

2. Overall API requirement

2.1. Possible API scheme

We support two kinds of API scheme for integration, in which "Scheme 1" is recommended. Their differences are described below:

2.1.1. Pattern 1 **(Recommended)**

We need to have 3 APIs from CSP.

- Fullset metadata API: will be used to get the initial dataset from CSP. It is better to have pagination logic for this API.
- Incremental metadata API: will be used to get the changed (added / updated / deleted) datanode within a specific time span (start / end time) after the first initial data retrieval.
- Single metadata API: will be used to cherry pick a single metadata node to refresh its information.

By using this way, we could get the updated metadata faster so that users on our platform could enjoy new contents in a timely manner.
More detailed information will be described in Section 3 for this pattern.

2.1.2. Pattern 2

We need to have 2 APIs from CSP.

- Fullset metadata API: will be used to get the initial dataset from CSP. If this API is just to let us download a big file, it should support resuming downloading from break point in case of poor network condition.
- Single metadata API: will be used to cherry pick a single metadata node to refresh its information.

Though the development work of this pattern is lower than pattern 1, their might be time lag on refreshing metadata source, which might cause displaying problem on our TV endpoints.

2.2. Supported Data Model

We support the following data models with different kind of structure:

- Single layer data model:
  Videos (including media types such as: movie, short_video, ugc(user_generated_content)),
  Sports(corresponding media type:sport),
  Apps(corresponding media type:app),
  Music(corresponding media type:music)
- Two-layer data model:
  Live_channels(corresponding media type:live_channel), which has channel => schedule

- Three-layer data model:
  TV_series(corresponding media type:tv_series), which has series => season => episode

2.3. Multi-language patterns for your metadata

If your metadata supports multiple metadata languages, it is recommended to split the data into different media by the metadata language being used.

```
"media_1": {
    "id": "xxx",
    "meta_language": "eng",
    "title": "xxx",
    ...
},
"media_1": {
    "id": "yyy",
    "meta_language": "fre",
    "title": "yyy",
    ...
},
...
```

Or you can deliver all necessary fields, such as title, subtitle, description, and other fields as a list format.

```
"media_1": {
    "id": "xxx",
    "title": [
        {
            "text": "Real Sociedad - Getafe",
            "meta_language": "eng"
        },
        {
            "text": "Real Sociedad - Getafe",
            "meta_language": "fre"
        }
    ],
    ...
},
…
```

3. Code samples for API Pattern 1

3.1. Full-set APIs for all media types

- Supported media_type see section 2.2 Supported Data Model

3.1.1. Get all Genres List

GET /{media_type}/genres/list.json
Supported media_type see section 2.2 Supported Data Model

Response body
If successful, the response body contains data with the following structure:

JSON representation

```
{
   "data": [
    {
      object (GenresEntity)
    }
  ],
  "errcode": string,
  "errmsg": string,
  "total": number
}
```

JSON example

```
{
    "data":[
        {
            "Id":"1",
            "media_type":"movie",
```

```
            "title":[
                {
                    "text":"Action",
                    "meta_language":"eng"
                },
                {
                    "text":"Aktion",
                    "meta_language":"ger"
                },
                {
                    "text":"アクション",
                    "meta_language":"jpn"
                }
            ]
        }
    ],
    "errcode":100000,
    "errmsg":"success",
    "total":20
}
```


3.2. APIs for single layer media types

- Single layer data model:
  Videos (including media types such as: movie, short_video, ugc(user_generated_content)),
  Sports(corresponding media type:sport),
  Apps(corresponding media type:app),
  Music(corresponding media type:music)

3.2.1. Get Content List

GET /{model_type}/list.json

Example:
GET /Videos/list.json
GET /Sports/list.json

GET /Apps/list.json
GET /Music/list.json

Query parameters

Parameter	Description
genre_id	String. (Optional) The genres to include in return results. If none are provided, all genres will be returned. Specifying the same genre multiple times has the same effect as specifying it once.
page	Number. (Optional) page number.
page_size	Number. (Optional) The maximum number of one page to return.

Response body

If successful, the response body contains data with the following structure:

JSON representation

```
{
  "data": [
    {
      object (Entity)
    }
  ],
  "errcode": string,
  "errmsg": string,
  "page": number,
  "page_size": number,
  "total": number
}
```

Note: the Entity depend on model_types.

JSON Example（Movie）


```
{
    "errcode":100000,
    "errmsg":"success",
    "page":1,
    "page_size":1,
    "total":11,
    "data":[
        {
            "id":"1",
            "media_type":"movie",
            "status":1,
            "create_time":"2018-08-20 05:30:01",
            "modify_time":"2019-09-26 01:00:03",
            "metadata_language":"eng",
            "title":"Best Lab Ever",
            "subheading":"Today Ryan is showing off his awesome base!",
            "description":"Today Ryan is showing off his awesome base!",
            "audio_language":{
                "ALL":[
                    "eng",
                    "fre"
                ]
            },
            "genre":"Action",
            "genre_id":"tgtg8000002340",
            "tags":"",
            "release_date":"2018-08-20",
            "video_type":1,
            "score":{
                "self":10
            },
```

```
"resolution":"SD,HD",
"duration":747,
"tryout_duration":0,
"streaming_rights":[
    "USA",
    "FRA"
],
"pricing_info":{
    "ALL":{
        "free":0
    }
},
"video_rating":{
    "USA":"G"
},
"playback_urls":{
    "ALL":{
        "TV":{
            "SD":{
                "jump_url":"https://xx",
                "streaming_url":"https://api.xx"
            },
            "HD":{
                "jump_url":"https://xx",
                "streaming_url":"https://api.xx"
            }
        },
        "Android":{
            "jump_url":"https://api.xx"
        },
        "iOS":{
            "jump_url":"https://api.xx"
        }
```

```
        }
    },
    "screenshots":null,
    "poster":[
        {
            "poster_ratio":"16:9",
            "poster_dimension":{
                "width":1920,
                "height":1080
            },
            "poster_type":"horizontal",
            "poster_url":"https://xx.jpg"
        },
        {
            "poster_ratio":"7:10",
            "poster_dimension":{
                "width":700,
                "height":1000
            },
            "poster_type":"vertical",
            "poster_url":"https://xx.jpg"
        },
        {
            "poster_ratio":"3:4",
            "poster_dimension":{
                "width":960,
                "height":720
            },
            "poster_type":"horizontal",
            "poster_url":"https://xx.jpg"
        },
        {
            "poster_ratio":"1:1",
```

```
                        "poster_dimension":{
                            "width":720,
                            "height":720
                        },
                        "poster_type":"horizontal",
                        "poster_url":"https://xx.jpg"
                    }
                ]
            }
        ]
}
```


3.2.2. Get Content Detail Batchly

GET /{model_type}/detail/list.json

Query parameters

Parameter	Description
ids	string. Required. multiple values separated by commas.

Response body
If successful, the response body contains data with the following structure:

JSON representation
{ "data": [{

```
      object (Entity)
    }
  ],
  "errcode": string,
  "errmsg": string
}
```

Note: the Entity depend on model_types.

JSON Example（Movie）


```
{
    "errcode":100000,
    "errmsg":"success",
    "data":[
        {
            "id":"1",
            "media_type":"movie",
            "status":0
        },
        {
            "id":"1",
            "media_type":"movie",
            "status":1,
            "create_time":"2018-08-20 05:30:01",
            "modify_time":"2019-09-26 01:00:03",
            "metadata_language":"eng",
            "title":"Best Lab Ever",
            "subheading":"Today Ryan is showing off his awesome base!",
            "description":"Today Ryan is showing off his awesome base!",
            "audio_language":{
                "ALL":[
                    "eng",
                    "fre"
```

```
        ]
    },
    "genre":"Action",
    "genre_id":"tgtg8000002340",
    "tags":"",
    "release_date":"2018-08-20",
    "video_type":1,
    "score":{
        "self":10
    },
    "resolution":"SD,HD",
    "duration":747,
    "tryout_duration":0,
    "streaming_rights":[
        "USA"
    ],
    "pricing_info":{
        "ALL":{
            "free":0
        }
    },
    "video_rating":{
        "USA":"G"
    },
    "playback_urls":{
        "ALL":{
            "TV":{
                "SD":{
                    "jump_url":"https://xx",
                    "streaming_url":"https://api.xx"
                },
                "HD":{
                    "jump_url":"https://xx",
```

```
                    "streaming_url":"https://api.xx"
                }
            },
            "Android":{
                "jump_url":"https://api.xx"
            },
            "iOS":{
                "jump_url":"https://api.xx"
            }
        }
    },
    "screenshots":null,
    "poster":[
        {
            "poster_ratio":"16:9",
            "poster_dimension":{
                "width":1920,
                "height":1080
            },
            "poster_type":"horizontal",
            "poster_url":"https://xx.jpg"
        },
        {
            "poster_ratio":"7:10",
            "poster_dimension":{
                "width":700,
                "height":1000
            },
            "poster_type":"vertical",
            "poster_url":"https://xx.jpg"
        },
        {
            "poster_ratio":"3:4",
```

```
                    "poster_dimension":{
                        "width":960,
                        "height":720
                    },
                    "poster_type":"horizontal",
                    "poster_url":"https://xx.jpg"
                },
                {
                    "poster_ratio":"1:1",
                    "poster_dimension":{
                        "width":720,
                        "height":720
                    },
                    "poster_type":"horizontal",
                    "poster_url":"https://xx.jpg"
                }
            ]
        }
    ]
}
```


3.2.3 Get Incremental Content List （return added / updated / deleted contents）

GET /{model_type}/change/list.json

Query parameters

Parameter	Description

start_time	Int. Required. The start time. UTC Unix Timestamp(seconds). Example: 1574939700
end_time	Int. Optional. The end time. UTC Unix Timestamp(seconds). Example: 1575026099 If none are provided, default value is current time.
page	Number. Optional. page number.
page_size	Number. Optional. The maximum number of one page to return.

Response body

If successful, the response body contains data with the following structure:

JSON representation

```
{
  "data": [
    {
       "Id":"111",
        "media_type":"movie",
      "update_type":"deleted"
    },
    {
       "Id":"222",
        "media_type":"movie",
      "update_type":"insert"
    },
    {
       "Id":"333",
        "media_type":"movie",
      "update_type":"update"
    }
  ],
  "errcode": string,
  "errmsg": string,
```

```
  "page": number,
  "page_size": number,
  "total": number
}
```

Note:
Supported media_type：”movie”,”short_video”,”ugc”
Supported `update_type`: ”insert”, ”update”, ”deleted”.

3.3. APIs for 2-layer media types (Live Channels)

- Two-layer data model:
  Live_channels(corresponding media type:live_channel), which has channel => schedule

3.3.1. Get Channel List

GET /channels/list.json

Query parameters

Parameter	Description
page	Number. Optional. page number.
page_size	Number. Optional. The maximum number of one page to return.

Response body
If successful, the response body contains data with the following structure:

JSON representation

```
{
  "data": [
    {
      object (ChannelEntity)
    }
  ],
  "errcode": string,
  "errmsg": string,
  "page": number,
  "page_size": number,
  "total": number
}
```

3.3.2. Get Schedule List by Channel Id

GET /schedule/list.json

Query parameters

Parameter	Description
channel_id	String. Required. channel unique identifier.
start_time	Int. (Optional). The start time. UTC Unix Timestamp(seconds). Example: 1574939700 If none are provided, default value is 0, should return full set datas.
end_time	Int. (Optional). The end time. UTC Unix Timestamp(seconds). Example: 1575026099 If none are provided, default value is current time.

page	Number. (Optional). page number.
page_size	Number.(Optional）. The maximum number of one page to return.

Response body
If successful, the response body contains data with the following structure:

JSON representation

```
{
  "data": [
    {
      object (ScheduleEntity)
    }
  ],
  "errcode": string,
  "errmsg": string,
  "page": number,
  "page_size": number,
  "total": number
}
```

3.4. APIs for 3-layer media types (TV Series)

- Three-layer data model:
  TV_series(corresponding media type:tv_series), which has series => season => episode

3.4.1. Get Series List

GET /TV_series/list.json

Query parameters

Parameter	Description
genre_id	String. Optional. The genres to include in return results. If none are provided, all genres will be returned. Specifying the same genre multiple times has the same effect as specifying it once.
page	Number. Optional. page number.
page_size	Number. Optional. The maximum number of one page to return.

Response body

If successful, the response body contains data with the following structure:

JSON representation

```
{
  "data": [
    {
      object (SeriesEntity)
    }
  ],
  "errcode": string,
  "errmsg": string,
  "page": number,
  "page_size": number,
  "total": number
}
```

JSON Example（TV Series）

```
{
    "data":[
```

```
    {
        "id":"1",
        "media_type":"tv_series",
        "program_type":"series",
        "title":"Mouse In The House",
        "metadata_language":"eng"
    },
    {
        "id":"2",
        "media_type":"tv_series",
        "program_type":"series",
        "title":"Cast Of Lab Rats",
        "metadata_language":"eng"
    },
    {
        "id":"3",
        "media_type":"tv_series",
        "program_type":"series",
        "title":"Do Something Awards",
        "metadata_language":"eng"
    },
    {
        "id":"4",
        "media_type":"tv_series",
        "program_type":"series",
        "title":"All News For Ariana Grande",
        "metadata_language":"eng"
    },
    {
        "id":"5",
        "media_type":"tv_series",
        "program_type":"series",
        "title":"Good Luck Charlie Cast News",
```

```
            "metadata_language":"eng"
        }
    ],
    "errcode":100000,
    "errmsg":"success",
    "page":1,
    "page_size":5,
    "total":20
}
```


Note: If media type is “tv_series”, the “Get Content List” API can just return series detail list. And then can get season and episode details by “Get Season List By Series Id” and “Get Episode List By Season Id”.

3.4.2. Get Season List By Series Id

GET /seasons/list.json

Query parameters

Parameter	Description
ids	String. Required. series unique identifier. multiple values separated by commas.
page	Number. Optional. page number.
page_size	Number. Optional. The maximum number of one page to return.

Response body
If successful, the response body contains data with the following structure:

JSON representation

```
{
  "data": [
    {
      object (SeasonEntity)
    }
  ],
  "errcode": string,
  "errmsg": string,
  "page": number,
  "page_size": number,
  "total": number
}
```

JSON Example


```
{
    "errcode":100000,
    "errmsg":"success",
    "page":1,
    "page_size":5,
    "total":20,
    "data":[
        {
            "id":"ss1",
            "media_type":"tv_series",
            "program_type":"season",
            "series_id":"s1",
            "status":1,
            "create_time":"2011-05-31 06:58:29",
            "modify_time":"2019-09-30 04:12:35",
            "metadata_language":"eng",
            "season_number":1,
```

```
"title":"Mouse In The House Season 1",
"subheading":"Learn the wonders of science with your favorite Mouse!",
"description":"Learn the wonders of science with your favorite Mouse!",
"audio_language":[
    "eng"
],
"screenshots":null,
"poster":[
    {
        "poster_ratio":"16:9",
        "poster_dimension":{
            "width":1920,
            "height":1080
        },
        "poster_type":"horizontal",
        "poster_url":"https://xx.jpg"
    },
    {
        "poster_ratio":"7:10",
        "poster_dimension":{
            "width":700,
            "height":1000
        },
        "poster_type":"vertical",
        "poster_url":"https://xx.jpg"
    },
    {
        "poster_ratio":"3:4",
        "poster_dimension":{
            "width":960,
            "height":720
        },
        "poster_type":"horizontal",
```

```
            "poster_url":"https://xx.jpg"
        },
        {
            "poster_ratio":"1:1",
            "poster_dimension":{
                "width":720,
                "height":720
            },
            "poster_type":"horizontal",
            "poster_url":"https://xx.jpg"
        },
        {
            "poster_ratio":"4:3",
            "poster_dimension":{
                "width":720,
                "height":960
            },
            "poster_type":"vertical",
            "poster_url":"https://xx.jpg"
        }
    ],
    "genre":"Adventure",
    "genre_id":"tgtg8000000009",
    "sub_genre":"Fantasy",
    "sub_genres_id":"tgtg8000000002",
    "tags":"learning,science,physics,mouse,chemistry,chemicals,experiment",
    "score":{
        "self":10
    },
    "streaming_rights":[
        "GLOBAL"
    ],
    "pricing_info":{
```

```
        },
        "playback_urls":{
            "GLOBAL":{
                "TV":{
                    "SD":{
                        "jump_url":"https://xx"
                    },
                    "HD":{
                        "jump_url":"https://xx"
                    }
                },
                "Android":{
                    "jump_url":"https://api.xx"
                },
                "iOS":{
                    "jump_url":"https://api.xx"
                }
            }
        }
    },
    {
        "id":"ss2",
        "media_type":"tv_series",
        "program_type":"season",
        "series_id":"s1",
        "status":1,
        "create_time":"2011-05-31 06:58:29",
        "modify_time":"2019-09-30 04:12:35",
        "metadata_language":"eng",
        "season_number":2,
        "title":"Mouse In The House Season 2",
        "subheading":"Learn the wonders of science with your favorite Mouse!",
```

```
"description":"Learn the wonders of science with your favorite Mouse!",
"audio_language":[
    "eng"
],
"screenshots":null,
"poster":[
    {
        "poster_ratio":"16:9",
        "poster_dimension":{
            "width":1920,
            "height":1080
        },
        "poster_type":"horizontal",
        "poster_url":"https://xx.jpg"
    },
    {
        "poster_ratio":"7:10",
        "poster_dimension":{
            "width":700,
            "height":1000
        },
        "poster_type":"vertical",
        "poster_url":"https://xx.jpg"
    },
    {
        "poster_ratio":"3:4",
        "poster_dimension":{
            "width":960,
            "height":720
        },
        "poster_type":"horizontal",
        "poster_url":"https://xx.jpg"
    },
```

```
            {
                "poster_ratio":"1:1",
                "poster_dimension":{
                    "width":720,
                    "height":720
                },
                "poster_type":"horizontal",
                "poster_url":"https://xx.jpg"
            },
            {
                "poster_ratio":"4:3",
                "poster_dimension":{
                    "width":720,
                    "height":960
                },
                "poster_type":"vertical",
                "poster_url":"https://xx.jpg"
            }
        ],
        "genre":"Adventure",
        "genre_id":"tgtg8000000009",
        "sub_genre":"Fantasy",
        "sub_genres_id":"tgtg8000000002",
        "tags":"learning,science,physics,mouse,chemistry,chemicals,experiment",
        "score":{
            "self":10
        },
        "streaming_rights":[
            "GLOBAL"
        ],
        "pricing_info":{

        },
```

```
            "playback_urls":{
                "GLOBAL":{
                    "TV":{
                        "SD":{
                            "jump_url":"https://xx"
                        },
                        "HD":{
                            "jump_url":"https://xx"
                        }
                    },
                    "Android":{
                        "jump_url":"https://api.xx"
                    },
                    "iOS":{
                        "jump_url":"https://api.xx"
                    }
                }
            }
        ]
}
```


3.4.3. Get Episode List By Season Id

GET /episodes/list.json

Query parameters

Parameter	Description

ids	String. Required. season unique identifier, multiple values separated by commas.
page	Number. Optional. page number.
page_size	Number. Optional. The maximum number of one page to return.

Response body
If successful, the response body contains data with the following structure:

JSON representation

```
{
  "data": [
    {
     object (EpisodeEntity)
    }
  ],
  "errcode": string,
  "errmsg": string,
  "page": number,
  "page_size": number,
  "total": number
}
```

JSON Example

```
{
    "errcode":100000,
```

```
"errmsg":"success",
"page":1,
"page_size":5,
"total":20,
"data":[
    {
        "id":"e1",
        "media_type":"tv_series",
        "program_type":"episode",
        "status":1,
        "create_time":"2011-06-01 06:59:23",
        "modify_time":"2019-09-25 11:59:25",
        "metadata_language":"eng",
        "title":"Lemon Battery",
        "subheading":"Mouse and friends build a battery using a lemon.",
        "description":"Mouse and friends build a battery using a lemon.",
        "audio_language":[
            "eng"
        ],
        "genre":"Adventure",
        "genre_id":"1",
        "sub_genre":"Fantasy",
        "sub_genres_id":"22",
        "release_date":"2011-06-01",
        "video_type":1,
        "score":{
            "self":10
        },
        "resolution":"SD",
        "duration":182,
        "tryout_duration":0,
        "streaming_rights":[
            "GLOBAL"
```

```
],
"pricing_info":{
    "GLOBAL":{
        "free":0
    }
},
"video_rating":{
    "USA":"TV-Y"
},
"playback_urls":{
    "GLOBAL":{
        "TV":{
            "SD":{
                "jump_url":"https://xx",
                "streaming_url":"https://api.xx"
            },
            "HD":{
                "jump_url":"https://xx",
                "streaming_url":"https://api.xx"
            }
        },
        "Android":{
            "jump_url":"https://api.xx"
        },
        "iOS":{
            "jump_url":"https://api.xx"
        }
    }
},
"screenshots":null,
"poster":[
    {
        "poster_ratio":"16:9",
```

```
            "poster_dimension":{
                "width":1920,
                "height":1080
            },
            "poster_type":"horizontal",
            "poster_url":"https://xx.jpg"
        },
        {
            "poster_ratio":"7:10",
            "poster_dimension":{
                "width":700,
                "height":1000
            },
            "poster_type":"vertical",
            "poster_url":"https://xx.jpg"
        },
        {
            "poster_ratio":"3:4",
            "poster_dimension":{
                "width":960,
                "height":720
            },
            "poster_type":"horizontal",
            "poster_url":"https://xx.jpg"
        },
        {
            "poster_ratio":"1:1",
            "poster_dimension":{
                "width":720,
                "height":720
            },
            "poster_type":"horizontal",
            "poster_url":"https://xx.jpg"
```

```
            }
        ],
        "season_id":"tgtg9000003168",
        "episode_number":1
    },
    {
        "id":"e2",
        "media_type":"tv_series",
        "program_type":"episode",
        "status":1,
        "create_time":"2011-06-01 06:59:23",
        "modify_time":"2019-09-25 11:59:25",
        "metadata_language":"eng",
        "title":"Rainbow Milk",
        "subheading":"Mouse and friends make rainbow milk.",
        "description":"Mouse and friends make rainbow milk.",
        "audio_language":[
            "eng"
        ],
        "genre":"Adventure",
        "genre_id":"tgtg8000000009",
        "sub_genre":"Fantasy",
        "sub_genres_id":"tgtg8000000002",
        "release_date":"2011-06-01",
        "video_type":1,
        "score":{
            "self":10
        },
        "resolution":"SD",
        "duration":186,
        "tryout_duration":0,
        "streaming_rights":[
            "GLOBAL"
```

```
],
"pricing_info":{
    "GLOBAL":{
        "free":0
    }
},
"video_rating":{
    "USA":"TV-Y"
},
"playback_urls":{
    "GLOBAL":{
        "TV":{
            "SD":{
                "jump_url":"https://xx",
                "streaming_url":"https://api.xx"
            },
            "HD":{
                "jump_url":"https://xx",
                "streaming_url":"https://api.xx"
            }
        },
        "Android":{
            "jump_url":"https://api.xx"
        },
        "iOS":{
            "jump_url":"https://api.xx"
        }
    }
},
"screenshots":null,
"poster":[
    {
        "poster_ratio":"16:9",
```

```
            "poster_dimension":{
                "width":1920,
                "height":1080
            },
            "poster_type":"horizontal",
            "poster_url":"https://xx.jpg"
        },
        {
            "poster_ratio":"7:10",
            "poster_dimension":{
                "width":700,
                "height":1000
            },
            "poster_type":"vertical",
            "poster_url":"https://xx.jpg"
        },
        {
            "poster_ratio":"3:4",
            "poster_dimension":{
                "width":960,
                "height":720
            },
            "poster_type":"horizontal",
            "poster_url":"https://xx.jpg"
        },
        {
            "poster_ratio":"1:1",
            "poster_dimension":{
                "width":720,
                "height":720
            },
            "poster_type":"horizontal",
            "poster_url":"https://xx.jpg"
```

```
                    }
                ],
                "season_id":"ss1",
                "episode_number":2
            }
        ]
}
```


3.4.4. Get Incremental Content List （return new added、updated、deleted contents）

GET /TV_Series/change/list.json

Query parameters

Parameter	Description
start_time	Int. Required. The start time. UTC Unix Timestamp(seconds). Example: 1574939700
end_time	Int. Optional. The end time. UTC Unix Timestamp(seconds). Example: 1575026099 If none are provided, default value is current time.
page	Number. Optional. page number.
page_size	Number. Optional. The maximum number of one page to return.

Response body
If successful, the response body contains data with the following structure:

JSON representation

```
{
  "data": [
    {
       "id":"111",
       "media_type":"tv_series",
      "program_type":"seasons",
      "update_type":"deleted"
    }
  ],
  "errcode": string,
  "errmsg": string,
  "page": number,
  "page_size": number,
  "total": number
}
```

Note:

（1）Supported program_type:“series”、“seasons” and “episodes”.

(2) Supported `update_type`:”insert”、”update”、”deleted”.

3.2.2. Get TV Series Content Detail Batchly

GET /TV_Series/{program_type}/detail/list.json

Query parameters

Parameter	Description
ids	string. Required. multiple values separated by commas.

Response body

If successful, the response body contains data with the following structure:

JSON representation

```
{
  "data": [
    {
      object (Entity)
    }
  ],
  "errcode": string,
  "errmsg": string
}
```

Note: the Entity depend on model_types.

JSON Example（series）

```
{
"errcode":100000,
"errmsg":"success",
"data":[
    {
        "id":"1",
        "media_type":"tv_series",
        "program_type":"series",
        "title":"Mouse In The House",
        "metadata_language":"eng"
    },
    {
        "id":"2",
        "media_type":"tv_series",
        "program_type":"series",
        "title":"Cast Of Lab Rats",
```

```
            "metadata_language":"eng"
        }
    ]
}
```

4. Field definition for each media type

4.1. Genres

Field	Description	Format	Mandatory	Sample value
id	unique identifier	string	Yes	
title	title	string	Yes	
metadata_language	3 chars language code	string	Yes	eng
media_type	media type. supported values see section 2.2 Supported Data Model	string	Yes	“movie”

4.2. Videos (including movies, short videos, and UGC)

Field	Description	Format	Mandatory	Sample value
id	unique identifier	string	Yes	

status	0-offline；1-online	int	Yes	1
media_type	media type. supported values: movie、short_video、ugc	string	Yes	“movie”
online_time	Online time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
offline_time	Offline time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
create_time	Create time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
modify_time	update time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
metadata_language	3 chars language code	string	Yes	eng
title	title	string	Yes	
subheading	subheading	string		
description	content description	string	Yes	
streaming_rights	the countries for which the program has streaming rights. 3 chars country code should be used. If the target content is globally available, please use “ALL” here.	list<string>	Yes	["DEU", "AUT"]
audio_ language	Audio language related to region. 3 chars language code	map	Yes	{ "DEU":["eng", "fre"] }

				NOTE:If audio language is not relevant to the region, as follows: { "ALL":["eng", "fre"] }
caption_language	subtitle or caption language in 3 chars language code	list<string>		{ "DEU":["eng", "fre"] } NOTE:If caption language is not relevant to the region, as follows: { "ALL":["eng", "fre"] }

poster	Each poster needs to include: Poster_url , Poster_ratio (16:9 or 2:3 or 4:3 or 3:4 or 1:1) , Poster_dimension {width , height} , Poster_type (“horizontal” or ”vertical” or “square”). Please provide different ratio posters as many as possible, and at least one for each ratio.	list(map)	Yes	<pre>[{ "poster_url":"http://xxx.yyy.zzz.webp", "poster_ratio":"3:4", "poster_dimensions":{"height":840, "width":560}, "poster_type":"horizontal" }, ...]</pre>
screenshots	Each screenshot needs to include: Screenshot_url Screenshot_ratio (16:9 or 2:3 or 4:3 or 3:4 or 1:1) Screenshot_dimensions {width , height} Screenshot_type (“horizontal” or ”vertical” or “square”). Please provide different ratio posters as many as possible, and at least one for each ratio.	list(map)		<pre>[{ "screenshot_url":"http://xxx.yyy.zzz.webp", "screenshot_ratio":"3:4", "screenshot_dimensions":{"height":840, "width":560}, "screenshot_type":"horizontal" }, ...]</pre>
genre	major genre. We need to get your full genre list to do the internal mapping.	string	Yes	TV
genre_id	The content major genres id. We need this for more accurate mapping.	string		
actors	If it's a variety show, the field is filled in as the resident guests. VIP actors should come to the front in the list.	list<string>	Yes	<pre>["Tom Cruise", ...]</pre>
directors	Main directors should come to the front in the list.	list<string>	Yes	<pre>["Luc Besson", ...]</pre>
tags	Important ones should come to the front in the list.	list<string>	Yes	<pre>["family", "kids", ...]</pre>

release_date	The publish time，yyyy-MM-dd	string	Yes	2018-01-01
video_type	1-VOD, 2-Cut, 3-Trailer;	int	Yes	1
score	Score (maximum is 10)	json	Yes	{"self":8.0, "IMDB":9.4}
resolution	Resolution supported, such as [SD,HD,4K,etc...]	list<string>	Yes	["SD","HD"]
duration	video length in seconds	int	Yes	5280
tryout_duration	tryout length in seconds, 0 or blank means tryout is not supported	int		60
pricing_info	pricing information should contain TVOD and SVOD types in each country. **TVOD-RENT**: use rent time as the key (24h，7d，1m) and pricing as the value. **TVOD-EST**(electrical sell-through): use est and key and pricing as value. **SVOD**: use SVOD as key and subscription package id as the value. **Free**: use free as key and 1 or 0 as value, in which 1:Free and 0: FreeWithAds Note-1: Please also provide another table to describe your subscription packages. Note-2: If the pricing_info available for all regions, the region code should be “ALL”.	map	Yes	{ "DEU": { "free": 1 or 0, "24h" : {"SD": 5.99, "HD": 6.99, ... }, "7d" : {"SD": 5.99, "HD": 6.99, ...}, "est": {"SD": 5.99, "HD": 6.99, ...}, "svod": ["123454545", "123454545", "123454545", ...] }, ... }
video_rating	Rating mark within each targeting country.	map	Yes	{"USA":"PG", "GBR":"age_12_and_over"}

playback_urls	playback url for different platforms and integration patterns in each country. **Platform**: TV, iOS, Android. And iOS/Android are for mobile playback. **deep_linking_url**: deep-linking pattern that will open CSP's app to play content. **streaming_url**: this will open VIDAA's own player to play content. **requesting_url**: VIDAA will use this url to get the real streaming url from CSP's server. Note: If the playback_urls available for all regions, the region code should be "ALL".	map	Yes	<pre>{ "USA": { "TV": { "SD": { "deep_linking_url": http://xxx.yyy.zzz, "streaming_url": http://xxx.yyy.zzz, "requesting_url": http://xxx.yyy.zzz }, "HD": {.....}, "4K": {.....} }, "Android": {......}, "iOS": {......} }, "GBR": {......}, ... }</pre>

4.3. TV Series (Drama, Entertainment, Comic etc..)

4.3.1. Series

Field	Description	Format	Mandatory	Sample value
id	unique identifier	string	Yes	
media_type	media type	string	Yes	"tv_series"
program_type	the type of program	string	Yes	"series"
title	title	string	Yes	
metadata_language	3 chars language code	string	Yes	eng

4.3.2. Seasons

Field	Description	Format	Mandatory	Sample value
id	unique identifier	string	Yes	
media_type	media type	string	Yes	“tv_series”
program_type	the type of program	string	Yes	“season”
series_id	key information to identify series	string	Yes	
status	0-offline；1-online	int	Yes	1
online_time	Online time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
offline_time	Offline time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
create_time	Create time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
modify_time	update time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
metadata_language	3 chars language code	string	Yes	eng
season_number	season number	int		
title	title	string	Yes	
subheading	subheading	string		

description	content description	string	Yes	
streaming_rights	the countries for which the program has streaming rights. 3 chars country code should be used. If the target content is globally available, please use “ALL” here.	list<string>	Yes	`["DEU", "AUT"]`
audio_ language	Audio language related to region. 3 chars language code	map	Yes	`{` `"DEU":[` `"eng",` `"fre"` `]` `}` `NOTE:If audio language is not relevant to the region, as follows:` `{` `"ALL":[` `"eng",` `"fre"` `]` `}`
caption_language	subtitle or caption language in 3 chars language code	list<string>		`{` `"DEU":[` `"eng",` `"fre"` `]` `}`

				NOTE:If caption language is not relevant to the region, as follows: { "ALL":["eng", "fre"] }
poster	Each poster needs to include: Poster_url , Poster_ratio (16:9 or 2:3 or 4:3 or 3:4 or 1:1) , Poster_dimension {width , height} , Poster_type (“horizontal” or ”vertical” or “square”). Please provide different ratio posters as many as possible, and at least one for each ratio.	list(map)	Yes	[{ "poster_url":"http://xxx.yyy.zzz.webp", "poster_ratio":"3:4", "poster_dimensions":{"height":840, "width":560}, "poster_type":"horizontal" }, ...]
screenshots	Each screenshot needs to include: Screenshot_url Screenshot_ratio (16:9 or 2:3 or 4:3 or 3:4 or 1:1) Screenshot_dimensions {width , height} Screenshot_type (“horizontal” or ”vertical” or “square”). Please provide different ratio posters as many as possible, and at least one for each ratio.	list(map)		[{ "screenshot_url":"http://xxx.yyy.zzz.webp", "screenshot_ratio":"3:4", "screenshot_dimensions":{"height":840, "width":560}, "screenshot_type":"horizontal" }, ...]
genre	major genre. We need to get your full genre list to do the internal mapping.	string	Yes	TV

genre_id	The content major genres id. We need this for more accurate mapping.	string		
actors	Main directors should come to the front in the list.	list<string>	Yes	["Tom Cruise", ...]
directors	Important ones should come to the front in the list.	list<string>	Yes	["Luc Besson", ...]
hosts	Main hosts should come to the front in the list. Cannot be blank, if the video is a variety program.	list<string>		
tags	Important ones should come to the front in the list.	list<string>	Yes	["family", "kids", ...]
score	Score (maximum is 10)	JSON	Yes	{"self":8.0, "IMDB":9.4}
video_rating	Rating mark within each targeting country.	list(map)	Yes	{"USA":"PG", "GBR":"age_12_and_over"}
pricing_info	pricing information should contain TVOD and SVOD types in each country. **TVOD-RENT**: use rent time as the key (24h, 7d, 1m) and pricing as the value. **TVOD-EST**(electrical sell-through): use est and key and pricing as value. **SVOD**: use SVOD as key and subscription package id as the value. **Free**: use free as key and 1 or 0 as value, in which 1:Free and 0: FreeWithAds Note-1: Please also provide another table to describe your subscription packages. Note-2: If the pricing_info available for all regions, the region code should be “ALL”.	list(map)	Yes	{ "DEU": { "free": 1 or 0, "24h" : {"SD": 5.99, "HD": 6.99, ... }, "7d" : {"SD": 5.99, "HD": 6.99, ...}, "est": {"SD": 5.99, "HD": 6.99, ...}, "svod": ["123454545", "123454545", "123454545", ...] }, ... }

playback_urls	playback url for different platforms and integration patterns in each country. **Platform**: TV, iOS, Android. And iOS/Android are for mobile playback. **deep_linking_url**: deep-linking pattern that will open CSP's app to play content. **streaming_url**: this will open VIDAA's own player to play content. **requesting_url**: VIDAA will use this url to get the real streaming url from CSP's server. Note: If the playback_urls available for all regions, the region code should be "ALL".	list(map)	Yes	<pre>{ "USA": { "TV": { "SD": { "deep_linking_url": http://xxx.yyy.zzz, "streaming_url": http://xxx.yyy.zzz, "requesting_url": http://xxx.yyy.zzz }, "HD": {.....}, "4K": {.....} }, "Android": {......}, "iOS": {......} }, "GBR": {......}, ... }</pre>

4.3.3. Episodes

Field	Description	Format	Mandatory	Sample value
id	unique identifier	string	Yes	
series_id	key information to identify series	string	Yes	
season_id	key information to identify season	string	Yes	
media_type	media type	string	Yes	"tv_series"
program_type	the type of program	string	Yes	"episode"

status	0-offline；1-online	int	Yes	1
online_time	Online time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
offline_time	Offline time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
create_time	Create time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
modify_time	update time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
metadata_language	3 chars language code, ISO 639-2 B	string	Yes	eng
episode_number	episode number	int	Yes	
title	title	string	Yes	
streaming_rights	the countries for which the program has streaming rights. 3 chars country code should be used. If the target content is globally available, please use “ALL” here.	list<string>	Yes	["DEU", "AUT"]
audio_ language	Audio language related to region. 3 chars language code	map	Yes	{ "DEU":["eng", "fre"] } NOTE:If audio language is not relevant to the region, as follows: { "ALL":[

				"eng", "fre"] }
caption_language	subtitle or caption language in 3 chars language code	list<string>		{ "DEU":["eng", "fre"] } NOTE:If caption language is not relevant to the region, as follows: { "ALL":["eng", "fre"] }

poster	Each poster needs to include: Poster_url , Poster_ratio (16:9 or 2:3 or 4:3 or 3:4 or 1:1) , Poster_dimension {width , height} , Poster_type (“horizontal” or ”vertical” or “square”). Please provide different ratio posters as many as possible, and at least one for each ratio.	list(map)	Yes	<pre>[{ "poster_url":"http://xxx.yyy.zzz.webp", "poster_ratio":"3:4", "poster_dimensions":{"height":840, "width":560}, "poster_type":"horizontal" }, ...]</pre>
screenshots	Each screenshot needs to include: Screenshot_url Screenshot_ratio (16:9 or 2:3 or 4:3 or 3:4 or 1:1) Screenshot_dimensions {width , height} Screenshot_type (“horizontal” or ”vertical” or “square”). Please provide different ratio posters as many as possible, and at least one for each ratio.	list(map)		<pre>[{ "screenshot_url":"http://xxx.yyy.zzz.webp", "screenshot_ratio":"3:4", "screenshot_dimensions":{"height":840, "width":560}, "screenshot_type":"horizontal" }, ...]</pre>
release_date	The publish time，yyyy-MM-dd	string	Yes	2018-01-01
video_type	1-VOD, 2-Cut, 3-Trailer;	int	Yes	1
resolution	Resolution supported, such as [SD,HD,4K,etc...]	list<string>	Yes	["SD","HD"]
duration	video length in seconds	int	Yes	5280
tryout_duration	tryout length in seconds, 0 or blank means tryout is not supported	int		60

pricing_info	pricing information should contain TVOD and SVOD types in each country. **TVOD-RENT**: use rent time as the key (24h, 7d, 1m) and pricing as the value. **TVOD-EST**(electrical sell-through): use est and key and pricing as value. **SVOD**: use SVOD as key and subscription package id as the value. **Free**: use free as key and 1 or 0 as value, in which 1:Free and 0: FreeWithAds Note-1: Please also provide another table to describe your subscription packages. Note-2: If the pricing_info available for all regions, the region code should be “ALL”.	list(map)	Yes	<pre>{ "DEU": { "free": 1 or 0, "24h" : {"SD": 5.99, "HD": 6.99, ... }, "7d" : {"SD": 5.99, "HD": 6.99, ...}, "est": {"SD": 5.99, "HD": 6.99, ...}, "svod": ["123454545", "123454545", "123454545", ...] }, ... }</pre>
playback_urls	playback url for different platforms and integration patterns in each country. **Platform**: TV, iOS, Android. And iOS/Android are for mobile playback. **deep_linking_url**: deep-linking pattern that will open CSP's app to play content. **streaming_url**: this will open VIDAA's own player to play content. **requesting_url**: VIDAA will use this url to get the real streaming url from CSP’s server. Note: If the playback_urls available for all regions, the region code should be “ALL”.	list(map)	Yes	<pre>{ "USA": { "TV": { "SD": { "deep_linking_url": http://xxx.yyy.zzz, "streaming_url": http://xxx.yyy.zzz, "requesting_url": http://xxx.yyy.zzz }, "HD": {.....}, "4K": {.....} }, "Android": {......}, "iOS": {......} }, "GBR": {......}, ... }</pre>

4.4. Sports

Field	Description	Format	Mandatory	Sample value
id	unique identifier	string	Yes	
status	0-offline；1-online	int	Yes	1
media_type	media type	string	Yes	“sport”
online_time	Online time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
offline_time	Offline time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
create_time	Create time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
modify_time	update time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
metadata_language	3 chars language code, ISO 639-2 B	string	Yes	eng
title	title	string	Yes	
subheading	subheading	string		
description	content description	string	Yes	
streaming_rights	the countries for which the program has streaming rights. 3 chars country code should be used. If the target content is globally available, please use “ALL” here.	list<string>	Yes	["DEU", "AUT"]
audio_ language	Audio language related to region. 3 chars language code, ISO 639-2 B	map	Yes	{ "DEU":["eng",

				"fre"] } NOTE:If audio language is not relevant to the region, as follows: { "ALL":["eng", "fre"] }
poster	Each poster needs to include: Poster_url , Poster_ratio (16:9 or 2:3 or 4:3 or 3:4 or 1:1) , Poster_dimension {width , height} , Poster_type (“horizontal” or ”vertical” or “square”). Please provide different ratio posters as many as possible, and at least one for each ratio.	list(map)	Yes	[{ "poster_url":"http://xxx.yyy.zzz.webp", "poster_ratio":"3:4", "poster_dimensions":{"height":840, "width":560}, "poster_type":"horizontal" }, ...]

screenshots	Each screenshot needs to include: Screenshot_url Screenshot_ratio (16:9 or 2:3 or 4:3 or 3:4 or 1:1) Screenshot_dimensions {width , height} Screenshot_type (“horizontal” or ”vertical” or “square”). Please provide different ratio posters as many as possible, and at least one for each ratio.	list(map)		[{ "screenshot_url":"http://xxx.yyy.zzz.webp", "screenshot_ratio":"3:4", "screenshot_dimensions":{"height":840, "width":560}, "screenshot_type":"horizontal" }, ...]
genre	Sports genre. We need to get your full genre list to do the internal mapping.	string	Yes	Basketball
genre_id	The content major genres id. We need this for more accurate mapping.	string		
league	Tournament or league. We need to get your full league list to do the internal mapping.	string	Yes	NBA
league_id	Tournament or league id. We need this for more accurate mapping.	string	Yes	
participants	Participating team or player's name and id. We need to get your full team or player list to do the internal mapping.	list(map)	Yes	{ "ids_xxx":"Atlanta Hawks", "ids_yyy":"Boston Celtics", ... }
tags	Important ones should come to the front in the list.	list<string>	Yes	["Boston", "NBA", ...]
show_type	1-upcoming, 2-live, 3-catchup;	int	Yes	1
scheduled_start_time	live event scheduled start time yyyy-MM-dd HH:mm:ss in UTC	string		
scheduled_end_time	live event scheduled end time yyyy-MM-dd HH:mm:ss in UTC	string		

available_start_time	actual start time yyyy-MM-dd HH:mm:ss in UTC	string	Yes	
avialable_end_time	actual end time yyyy-MM-dd HH:mm:ss in UTC	string	Yes	
resolution	Resolution supported, such as [SD,HD,4K,etc...]	list<string>	Yes	["SD","HD"]
duration	video length in seconds	int	Yes	5280
tryout_duration	tryout length in seconds, 0 or blank means tryout is not supported	int		60
pricing_info	pricing information should contain TVOD and SVOD types in each country. **TVOD-RENT**: use rent time as the key (24h，7d，1m) and pricing as the value. **TVOD-EST**(electrical sell-through): use est and key and pricing as value. **SVOD**: use SVOD as key and subscription package id as the value. **Free**: use free as key and 1 or 0 as value, in which 1:Free and 0: FreeWithAds Note-1: Please also provide another table to describe your subscription packages. Note-2: If the pricing_info available for all regions, the region code should be “ALL”.	list(map)	Yes	{ "DEU": { "free": 1 or 0, "24h" : {"SD": 5.99, "HD": 6.99, ... }, "7d" : {"SD": 5.99, "HD": 6.99, ...}, "est": {"SD": 5.99, "HD": 6.99, ...}, "svod": ["123454545", "123454545", "123454545", ...] }, ... }
playback_urls	playback url for different platforms and integration patterns in each country. **Platform**: TV, iOS, Android. And iOS/Android are for mobile playback. **deep_linking_url**: deep-linking pattern that will open CSP's app to play content. **streaming_url**: this will open VIDAA's own player to play	list(map)	Yes	{ "USA": { "TV": { "SD": { "deep_linking_url": http://xxx.yyy.zzz, "streaming_url": http://xxx.yyy.zzz, "requesting_url": http://xxx.yyy.zzz }, "HD": {.....},

	content. **requesting_url**: VIDAA will use this url to get the real streaming url from CSP's server. Note: If the playback_urls available for all regions, the region code should be "ALL".			<pre>"4K": {.....} }, "Android": {......}, "iOS": {......} }, "GBR": {......}, ... }</pre>

4.5. Apps

Field	Description	Format	Mandatory	Sample value
id	unique identifier	string	Yes	
status	0-offline；1-online	int	Yes	1
media_type	media type	string	Yes	"app"
online_time	Online time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
offline_time	Offline time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
create_time	Create time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
modify_time	update time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
metadata_language	3 chars language code, ISO 639-2 B	string	Yes	eng
app_name	app name	string	Yes	
version	version	string		
description	app description	string	Yes	

icon	Both 16:9 and 1:1 size are needed. Poster_url , Poster_ratio (16:9 and 1:1) , Poster_dimension {width , height} , Poster_type (“horizontal” or “square”). Please provide different ratio posters as many as possible, and at least one for each ratio.	list(map)	Yes	[{ "poster_url":"http://xxx.yyy.zzz.webp", "poster_ratio":"16:9", "poster_dimensions":{"height":270, "width":480}, "poster_type":"horizontal" }, ...]
screenshots	Small screenshots and big screenshots are needed. Screenshot_url Screenshot_ratio (16:9) Screenshot_dimensions {width , height} Screenshot_type (“horizontal”). Please provide different ratio posters as many as possible, and at least one for each ratio.	list(map)		[{ "screenshot_url":"http://xxx.yyy.zzz.webp", "screenshot_ratio":"16:9", "screenshot_dimensions":{"height":1080, "width":1920}, "screenshot_type":"horizontal" }, ...]
developer	developer information	string	Yes	
regions	the countries for which the app can be used. 3 chars country code should be used. If the target content is globally available, please use “ALL” here.	list<string>	Yes	["DEU", "AUT"]
available_ language	languages that this app supports internally. 3 chars language code, ISO 639-2 B	map	Yes	{ "DEU":["eng", "fre"] } NOTE:If language is not relevant

				to the region, as follows: { "ALL":["eng", "fre"] }
genre	Major genre. We need to get your full genre list to do the internal mapping.	string	Yes	TV
genre_id	The content major genres id. We need this for more accurate mapping.	string		
pubdate	The publish time，yyyy-MM-dd	string	Yes	2018-01-01
latest_upgrade_date	Latest upgrade time, yyyy-MM-dd	string	Yes	2018-01-01
guidance	short msgs for guidance information	string		Guidance Suggested This app may include dynamic content.
score	Score (maximum is 10)	float		8.6
reviews_num	Number of comments	int		
devices	indicates the Hisense devices for which the app has been published.	list<string>	Yes	["BEX700","TV123"]
launch_url	URL to launch the app Note: If the launch_url available for all regions, the region code should be “ALL”.	string	Yes	{ "USA": http://xxx.yyy.zzz, "GBR": http://xxx.yyy.zzz ..... }

pricing	Payment info：0-free; 1-Payment is required in the application.	int	Yes	

4.6. Live Channels

4.6.1. Live Channel

Field	Description	Format	Mandatory	Sample value
id	unique identifier	string	Yes	
status	0-offline；1-online	int	Yes	1
media_type	media type	string	Yes	“live_channel”
program_type	program type	string	Yes	“channel”
online_time	Online time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
offline_time	Offline time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
create_time	Create time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
modify_time	update time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
metadata_language	3 chars language code, ISO 639-2 B	string	Yes	eng
title	Channel title	string	Yes	
description	short description	string	Yes	

genre	major genre. We need to get your full genre list to do the internal mapping.	string	Yes	TV
genre_id	The content major genres id. We need this for more accurate mapping.	string		
tags	Sorting has a priority order, important ones come to the front.	list<string>	Yes	["family", "kids", ...]
resolution	Resolution supported, such as [SD,HD,4K,etc...]	list<string>	Yes	["SD","HD"]
poster	Each poster needs to include: Poster_url , Poster_ratio (16:9 or 2:3 or 4:3 or 3:4 or 1:1) , Poster_dimension {width , height} , Poster_type (“horizontal” or ”vertical” or “square”). Please provide different ratio posters as many as possible, and at least one for each ratio.	list(map)	Yes	[{ "poster_url":"http://xxx.yyy.zzz.webp", "poster_ratio":"3:4", "poster_dimensions":{"height":840, "width":560}, "poster_type":"horizontal" }, ...]
screenshots	Each screenshot needs to include: Screenshot_url Screenshot_ratio (16:9 or 2:3 or 4:3 or 3:4 or 1:1) Screenshot_dimensions {width , height} Screenshot_type (“horizontal” or ”vertical” or “square”). Please provide different ratio posters as many as possible, and at least one for each ratio.	list(map)	Yes	[{ "screenshot_url":"http://xxx.yyy.zzz.webp", "screenshot_ratio":"3:4", "screenshot_dimensions":{"height":840, "width":560}, "screenshot_type":"horizontal" }, ...]
streaming_rights	the countries for which the program has streaming rights. 3 chars country code should be used. If the target content is globally available, please use “ALL” here.	list<string>	Yes	["DEU", "AUT"]

pricing_info	pricing information should contain TVOD and SVOD types in each country. **TVOD-RENT**: use rent time as the key (24h，7d，1m) and pricing as the value. **TVOD-EST**(electrical sell-through): use est and key and pricing as value. **SVOD**: use SVOD as key and subscription package id as the value. **Free**: use free as key and 1 or 0 as value, in which 1:Free and 0: FreeWithAds Note-1: Please also provide another table to describe your subscription packages. Note-2: If the pricing_info available for all regions, the region code should be “ALL”.	list(map)	Yes	{ "DEU": { "free": 1 or 0, "24h" : {"SD": 5.99, "HD": 6.99, ... }, "7d" : {"SD": 5.99, "HD": 6.99, ...}, "est": {"SD": 5.99, "HD": 6.99, ...}, "svod": ["123454545", "123454545", "123454545", ...] }, ... }
video_rating	Rating mark within each targeting country.	list(map)	Yes	{“USA”:”PG”, ”GBR”:”age_12_and_over”}
playback_urls	playback url for different platforms and integration patterns in each country. **Platform**: TV, iOS, Android. And iOS/Android are for mobile playback. **deep_linking_url**: deep-linking pattern that will open CSP's app to play content. **streaming_url**: this will open VIDAA's own player to play content. **requesting_url**: VIDAA will use this url to get the real streaming url from CSP’s server. Note: If the playback_urls available for all regions, the region code should be “ALL”.	list(map)	Yes	{ "USA": { "TV": { "SD": { "deep_linking_url": http://xxx.yyy.zzz, "streaming_url": http://xxx.yyy.zzz, "requesting_url": http://xxx.yyy.zzz }, "HD": {.....}, "4K": {.....} }, "Android": {......}, "iOS": {......} }, "GBR": {......}, ... }

4.6.2. Live_schedule

Field	Description	Format	Mandatory	Sample value
id	unique identifier	string	Yes	
media_type	media type	string	Yes	“live_channel”
program_type	program type	string	Yes	“schedule”
channel_id	related channel unique identifier	string	Yes	
title	program title	string	Yes	
description	program description	string	Yes	
actors	Main directors should come to the front in the list.	list<string>	Yes	`["Tom Cruise", ......]`
directors	Important ones should come to the front in the list.	list<string>	Yes	`["Luc Besson", ......]`
tags	Important ones should come to the front in the list.	list<string>	Yes	`["family", "kids", ......]`
start_time	schedule start time. UTC time in yyyy-MM-dd HH:mm:ss	string	Yes	2018-01-01 11:00:00
end_time	schedule end time. UTC time in yyyy-MM-dd HH:mm:ss	string	Yes	2018-01-01 12:00:00

4.7. Music

Field	Description	Format	Mandatory	Sample value
id	unique identifier	string	Yes	
status	0-offline；1-online	int	Yes	1
media_type	media type	string	Yes	“music”
online_time	Online time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
offline_time	Offline time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
create_time	Create time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
modify_time	update time. UTC time in yyyy-MM-dd HH:mm:ss	string		2018-01-01 12:00:00
metadata_language	3 chars language code, ISO 639-2 B	string	Yes	eng
music_type	1: single or 2: playlist	string	Yes	
playlist_type:	1: Album: soundtrack album 2: Playlist: list of soundtrack or music video 3: Collection: list of an artist or music genre/ranking/event	string	Yes if playlist	
title	Single: Name of this single music piece Album: Album name Playlist: Playlist name Collection: Artist/genre/ranking/event name	string	Yes	
description	short description	string	Yes	
artist	main artist name or “various artists”	string	Yes	

tags	Sorting has a priority order, important ones come to the front.	list<string>	Yes	["Pop", "R&R", "Maroon5", ...]
poster	Each poster needs to include: Poster_url , Poster_ratio (16:9 or 2:3 or 4:3 or 3:4 or 1:1) , Poster_dimension {width , height} , Poster_type (“horizontal” or ”vertical” or “square”). Please provide different ratio posters as many as possible, and at least one for each ratio.	list(map)	Yes	[{ "poster_url":"http://xxx.yyy.zzz.webp", "poster_ratio":"3:4", "poster_dimensions":{"height":840, "width":560}, "poster_type":"horizontal" }, ...]
number	The total number of music: 1 for single, n for playlist	int	Yes	
duration	The total length of music in seconds	int	Yes	
favorites	The total number of favorites added on CSP’s platform	int		
score	Score (maximum is 10)	float		
played	The total number of played counts on CSP’s platform	int		
quality	If media is VOD, then should be resolution like [SD,HD,4K,etc...] If media is soundtrack, then should be quality like [Standard, HQ, SQ, MQA, etc...]	list<string>	Yes	["SD","HD"]
streaming_rights	the countries for which the program has streaming rights. 3 chars country code should be used. If the target content is globally available, please use “ALL” here.	list<string>	Yes	["DEU", "AUT"]

playback_urls	playback url for different platforms and integration patterns in each country. **Platform**: TV, iOS, Android. And iOS/Android are for mobile playback. **deep_linking_url**: deep-linking pattern that will open CSP's app to play content. **streaming_url**: this will open VIDAA's own player to play content. **requesting_url**: VIDAA will use this url to get the real streaming url from CSP's server. Note: If the playback_urls available for all regions, the region code should be "ALL".	list(map)	Yes	<pre>{ "USA": { "TV": { "SD": { "deep_linking_url": http://xxx.yyy.zzz, "streaming_url": http://xxx.yyy.zzz, "requesting_url": http://xxx.yyy.zzz }, "HD": {.....}, "4K": {.....} }, "Android": {......}, "iOS": {......} }, "GBR": {......}, ... }</pre>

5. Appendix

5.1. Error code

Error Code	Error Msg
100000	success.
100002	Param invalid
100003	Permission denied
999999	Unknown error.

5.2. Image size

5.2.1. Posters, Icons and small screenshots

Ratio	Resolution
16:9	480 x 270
4:3	480 x 360
3:4	360 x 480
2:3	300 x 450
1:1	300 x 300

5.2.2. Full screenshots (only for Apps)

Ratio	Resolution
16:9	1920 x 1080