

Working with the Kaltura REACH API

In this article we cover the flows required when integrating with the Kaltura Reach API.

Kaltura REACH framework allows vendors to process Kaltura customer media and return enrichment assets, such as captions, transcripts, audio description, etc.

Vendors get access to special APIs ([entryVendorTask](#) - [Kaltura VPaaS API Documentation](#)) which allows them to periodically pull new REACH tasks (each Task will contain metadata required for the vendor to process the request), process them and finally, update the customer's Kaltura account with the resulting assets and/or metadata.

Vendors can choose to implement the integration using a Kaltura client library to add, list, update or delete objects from Kaltura. Kaltura supports a wide range of client libraries in different languages ([Native Client Libraries](#) - [Kaltura VPaaS API Documentation](#)) or the vendor can choose to use the REST API ([Kaltura VPaaS API Overview](#) - [Kaltura VPaaS API Documentation](#)).

REACH Vendor API - Basic Flow

General Guidelines

- **Encryption** - Requests have to be sent over HTTPS. Kaltura's clients require this when accessing their data or making API calls which result in new data.
- **Client tags** - This property is used by Kaltura to track which application issued which call. With REACH we make another use of this field; to ensure Kaltura has a way to determine the Task processing E2E, the vendor should follow the following standards:
 - If you use a client library, the client library exposes the ability to define a `clientTag` for each API request:
 - For not task-specific API calls, the `clientTag` should be set to be '`<default clientTag>_vendorName-vendorPartnerId`'. (`<default clientTag>` consists of the client library programming language and the library build date).
 - Example: 'php5:18-11-11_vendorName_12345'
 - For task-specific API calls, the Task ID should also be added to the `clientTag`.
 - Example, for a Task ID (9292) and vendor account id (12345), the resulting `clientTag` should be "php5:18-11-11_vendorName-12345-9292".
 - If you use direct API calls, then each API request should include a `clientTag` parameter within the request body that will include the vendor Application Version, for instance :
 - For not task-specific API calls, the `clientTag` should be set to be '`<default clientTag>_vendorName-vendorPartnerId`'. (`<default clientTag>` consists of the Vendor application Version Number).
 - Example: 'app:18-11-11_vendorName_12345'
 - For task-specific API calls, the Task ID should also be added to the `clientTag`.
 - Example, for a Task ID (9292) and vendor account id (12345), the resulting `clientTag` should be "app:18-11-11_vendorName-12345-9292".

Working with the Kaltura Reach API

Listing REACH Vendor Tasks and their parameters

- To execute API calls against Kaltura you need to first generate a valid KS ([How to Create a Kaltura Session](#) - [Kaltura VPaaS API Documentation](#)). The KS is generated using your dedicated account details. Important note: the KS does not need to be generated for each call. When generating the KS you have the ability to generate it with a specific expiry, the default is 24 hours.
 - The KS need to be generated with 'disableentitlement' privilege.
 - We strongly suggest provisioning an appToken for your vendor account and using that to spawn your API sessions. See [Application Tokens](#) - [Kaltura VPaaS API Documentation](#) for more information on appToken sessions. Note that you do not

need to generate a new app token each time you generate a KS; you can use an app token to generate a KS as long as your app token is valid.

- Also, you can find pre-compiled Kaltura API client libraries in a number of languages to help you get started.
- A vendor should list for pending jobs by sending the API [entryVendorTask.getJobs - Kaltura VPaaS API Documentation](#) and then gather details for the relevant jobs using [entryVendorTask.get - Kaltura VPaaS API Documentation](#). The result will be a list of entryVendorTask.

- **curl example** to list pending jobs

- Call [entryVendorTask.getJobs - Kaltura VPaaS API Documentation](#) to get a list of jobs that have been submitted by any customer users for your specific vendor services

```
curl -X POST https://www.kaltura.com/api_v3/service/reach_entryvendortask/action/getJobs \
-d "format=1" \
-d "ks=$KALTURA_SESSION" \
```

- In this call you need to use the app token generated from your Vendor PID's admin secret.
- This call will return a list of requested jobs for your Vendor account. By default the answer will only include tasks that are PENDING processing by the vendor. It is not possible to change the filter.
- The answer will return a list of jobs along with task IDs and access keys for each job. Make sure to store the task IDs and access keys until you have retrieved the assets that need to be processed and submitted your deliverables.
- Loop through the job objects in the response and get additional job details with [entryVendorTask.get - Kaltura VPaaS API Documentation](#), note that the vendor must add the `responseProfile[systemName]=reach_vendor` parameter to retrieve the REACH profile information of the job. The REACH profile information contains information such output format, metadata extraction, content deletion policy, Speaker Change Indication and profanity removal indication (see the ENUMS list)

```
curl -X POST https://www.kaltura.com/api_v3/service/reach_entryvendortask/action/get \
-d "format=1" \
-d "ks=$KALTURA_SESSION" \
-d "id=$TASK_ID" \
-d "responseProfile[systemName]=reach_vendor"
```

For each task call, you should use the access key as a value for the ks parameter, the access key is retrieved in the prior getJobs call.

- For each task, the vendor should change it `entryVendorTask[status]` to "Processing" using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#) before starting handling the task
- After the vendor processing is done, the vendor needs to update the `entryVendorTask[outputObjectId]` with the ID of the object that contains the output of the task.
 - If it is a captioning task, it would be the ID of the caption asset created using [captionAsset.add - Kaltura VPaaS API Documentation](#) (the ID is returned in the response).
- If for any reason the vendor can't process the task, it should set the task status to Error and set the `entryVendorTask[errDescription]` field (free text) with informative message about why the task couldn't be processed.
- The processing accuracy should be set on `entryVendorTask[accuracy]`.
 - If it is a captioning task, it also needs to be set on `captionAsset[accuracy]` when creating the caption using [captionAsset.add - Kaltura VPaaS API Documentation](#);

PHP example to list pending tasks and get the information for each one:

```
<?php
// $client initialization should be done before this line
// the response profile will make sure to also return the catalog item for this service and the customer reach profile details.
$responseProfile = new KalturaResponseProfileHolder();
$responseProfile->systemName = 'reach_vendor';
$client->setResponseProfile($responseProfile);
$reachPlugin = KalturaReachClientPlugin::get($client);
$entryVendorTaskFilter = new KalturaEntryVendorTaskFilter();
$entryVendorTaskFilter->createdAtGreaterThanOrEqual = time()-86400; // get all new jobs from last 24hrs
$tasks = $reachPlugin->entryVendorTask->getJobs($entryVendorTaskFilter);
```

- This will return a list of requested jobs for your Vendor account.

- The object type ResponseProfile set with the systemName attribute value to 'reach_vendor' tells the API to return additional information related to the Customer's REACH profile such as dictionaries, processing region, caption display settings, and other pertinent details.
- For each task, the vendor should change it `status` to "Processing" before starting handling the task
- After the vendor processing is done, the vendor needs to update the `KalturaEntryVendorTask->outputObjectId` with the ID of the object that contains the output of the task.
 - If it is a captioning task, it would be the `KalturaCaptionAsset->ID`.
- If for any reason the vendor can't process the task, it should set the task status to Error and set the `errDescription` field (free text) with informative message about why the task couldn't be processed.
- The processing accuracy should be set on `KalturaEntryVendorTask->accuracy`.
 - If it is a captioning task, it also needs to be set on `KalturaCaptionAsset->accuracy`;

For call and responses examples for retrieving jobs and tasks, please refer to the Appendix: [Retrieving jobs and retrieving a specific task | Kaltura Knowledge Center](#).

Example of a task object returned by Kaltura (in JSON format), includes the REACH vendor profile information

```
{
  "objects": [
    {
      "id": "15",
      "partnerId": 110,
      "vendorPartnerId": 108,
      "createdAt": 1521639688,
      "entryId": "0_o2z3458w",
      "status": 1,
      "reachProfileId": 1,
      "catalogItemId": 1,
      "accessKey": "OTNmNDE5Zjk2MDQxMTZkZWZhZjg1NTkzNTkxZDY5NDRjM2Y1ZmNiZWwzMTA7MTEwOzE1MjE3MjYwODg7MDsxNTIxNjM5Njg4LjQyMTM7O2VkaXQ6MF9vMnozNDU4dXZxZXRYb2x",
      "notes": "notes test",
      "dictionary": "Test Dictionary",
      "relatedObjects": {
        "reach_vendor_catalog_item": {
          "objects": [
            {
              "sourceLanguage": "Hebrew",
              "outputFormat": "1",
              "enableSpeakerId": true,
              "serviceType": 1,
              "serviceFeature": 1,
              "turnAroundTime": 1800,
              "objectType": "KalturaVendorCaptionsCatalogItem"
            }
          ],
          "totalCount": 1,
          "objectType": "KalturaVendorCatalogItemListResponse"
        }
      },
      "reach_vendor_profile": {
        "objects": [
          {
            "defaultSourceLanguage": "English",
            "defaultOutputFormat": 1,
            "enableMachineModeration": false,
            "enableHumanModeration": false,
            "autoDisplayMachineCaptionsOnPlayer": false,
            "autoDisplayHumanCaptionsOnPlayer": false,
            "enableMetadataExtraction": true,
            "enableSpeakerChangeIndication": false,
            "enableAudioTags": false,
            "enableProfanityRemoval": true,
            "contentDeletionPolicy": 1,
            "objectType": "KalturaReachProfile"
          }
        ],
        "totalCount": 1,
        "objectType": "KalturaReachProfileListResponse"
      }
    },
    {
      "objectType": "KalturaEntryVendorTask"
    }
  ],
  "totalCount": 1,
  "objectType": "KalturaEntryVendorTaskListResponse"
}
```

Extend accessKey Expiration

In case the vendor processing took longer than expected and the access key provided on the task was expired, Kaltura offers a way for the vendor to fetch a new access key.



Please note that the request for extending the access key is allowed only for tasks which are in processing state, anything else will result in an API exception.

To extend the access you can do the following call: <https://developer.kaltura.com/api-docs/service/entryVendorTask/action/extendAccessKey>

```
curl -X POST https://www.kaltura.com/api_v3/service/reach_entryvendortask/action/extendAccessKey \
-d "ks=$KALTURA_SESSION" \
-d "id=ENTRY_VENDOR_TASK_ID"
```

The returned object should contain the extended access key on the returned.

How to download video assets from Kaltura

There are many ways one can download assets from Kaltura but we have decided to use specific ones when it comes to vendor's fetching content from Kaltura. We support various customer use cases and REACH should work for all of them (customers who work with remote storage, encrypted content, regular Kaltura maintained content). Please follow these instructions when implementing your content fetching mechanism.

If the account has basic configuration and the content is stored within Kaltura we decided to utilize the fact that segmented HLS can be quickly concatenated and repackaged as an mp4 file without consuming a lot of bandwidth and CPU. Some of the capabilities offered by REACH may not require both the Video & Audio and fetching segmented HLS via Kaltura's packagers allows for a simple way to download both video & audio streams or only the audio stream. This will offer all the flexibility needed for you to select the streams you want to download and, in some cases, save unnecessary bandwidth consumption.

Please note that customers may define a desired **flavorParamsId** they wish external Vendor's to fetch from Kaltura. This **flavorParamsId** will be configured on the reach profile object. If this attribute is defined you should obey it and download from the entry's available **flavorAssets** only the flavor asset matching the requested **flavorParamsId**. This field will be comma separated and the order is important so please make sure to match an asset based on the user defined order. For example, if I defined **flavorParamIds** 1, 3, 5 and my entry has **flavorAssets** which were generated from **flavorParamIds** 2, 4, 5 then the asset matching **flavorParamsId** 5 should be fetched. Or if I defined **flavorParamIds** 1, 3, 5 and my entry has **flavorAssets** which were generated from **flavorParamIds** 1, 3, 7 then the asset matching **flavorParamsId** 1 should be fetched.

The `flavorParamsIds` returned in the REACH vendor profile is usually empty. Sometimes customers configure their client instance to return a specific value to indicate a preferred flavor they want the REACH vendor to use for download and processing. One use-case for example is a customer which has confidential videos where the video should not be viewed but they would still want a captioning or transcript, in such case they could indicate a specific flavor id in the `flavorParamsIds` to point to an audio-only flavor for the REACH vendor to download and process.

If the customer has not provided a preferred flavor then the REACH vendor can simply choose to download the source flavor or optionally can choose to download a compressed flavor for a faster and more efficient download.

How should it be achieved

STEP 1. Use Kaltura's API to fetch the entry information you need.

```
curl -X POST https://www.kaltura.com/api_v3/service/baseentry/action/getPlaybackContext \
-d "ks=OWRhZTUyNTI3ZGQ4YmZyZWwM3ZjWjY2RlYTAwODNmNzY2ZjY1ZWwM13w3MTk7MTE5OzE1NDcxMDk2MDc2Mj5xNTQ3MDIzMjA3LjYwMjQ7ZWxsYSS5aWRpY2hAa2FsdFdvYSS5bjb207KixkaXNhYXV0eD0=" \
-d "entryId=0_tpvajp5t" \
-d "contextDataParams[streamerType]=applehttp" \
-d "contextDataParams[ks]=OWRhZTUyNTI3ZGQ4YmZyZWwM3ZjWjY2RlYTAwODNmNzY2ZjY1ZWwM13w3MTk7MTE5OzE1NDcxMDk2MDc2Mj5xNTQ3MDIzMjA3LjYwMjQ7ZWxsYSS5aWRpY2hAa2FsdFdvYXV0eD0=" \
-d "contextDataParams[objectType]=KalturaPlaybackContextOptions"
```

Note the `contextDataParams[streamerType]=applehttp` indicates to use HLS

Note the KS field is set with the access key.

The most important fields for you will be:

- **Sources:** Contains the HLS .m3u8 Manifests URL's required to generate the HLS manifest.
- **flavorAssets:** Contains the entry's **flavorAssets** that can be streamed via HLS protocol.

In case the account setup does not support HLS playback (i.e call #1 returns without any valid sources) issue the same request

with `contextDataParams[streamerType]=http`. This will return in the sources section a URL for fetching the MP4 file directly.

In case the REACH profile is configured with specific **flavorParamsIds** use the **flavorAssets** section to identify which flavor asset you need to use. Each flavor asset will have an attribute called **flavorParamsId** which should be used for the match.

In any other case use the asset info such as **width, height, bitrate** information to select which flavor is the one you want to fetch.

STEP 2. From the **sources** section of the first call take the `url` field value (should be a link to the playManifest API) and fetch its content.

Don't forget to concatenate the tasks `accessKey` (From the response to [entryVendorTask.get](#) - Kaltura VPaaS API Documentation) to avoid any access restrictions to the entry and make sure the `"/flavorIds/"` in the URL contain only the flavor ID you wish to fetch. It should look like this:

```
https://cdnapisec.kaltura.com/p/$KALTURA_PARTNER_ID/sp/$KALTURA_PARTNER_ID00/playManifest/entryId/$KALTURA_ENTRY_ID/flavorIds/$KALTURA_HL
$KALTURA_HLS_FLAVOR_ID = FlavorId selected in step 2 $KALTURA_SESSION = accessKey from the entryVendorTask object.
```

OR in case of direct file download over HTTP:

```
https://cdnapisec.kaltura.com/p/$KALTURA_PARTNER_ID/sp/$KALTURA_PARTNER_ID00/playManifest/entryId/$KALTURA_ENTRY_ID/flavorIds/$SELECTED_F
```

STEP 3. In case HLS is supported the response of the `contextDataParams[streamerType]=applehttp` playManifest request should be a valid m3u8 manifest which contains a "serveFlavor" URL.

This "serveFlavor" URL points to the Kaltura packager or in case of remote storage should point to the customer's packager.

From the manifest parse the segment URL and use `FFmpeg` to fetch the segments and repackage the file into a file you can work with. This is an example command for concatenating the segments into an mp4 file:

```
ffmpeg -i "https://cdnapisec.kaltura.com/hls/p/$PARTNER_ID/sp/$PARTNER_ID00/serveFlavor/entryId/$ENTRY_ID/v/XXX/ev/XXX/flavorId/$FLAVOR_ASSET_ID/name/a.mp4/index.m3u8" -c copy -bsf:a i
```

To select which stream you want to fetch (Video, Audio || Both) follow the following documentation page <https://github.com/kaltura/nginx-vod-module#url-path-parameters>.

In case direct file download over HTTP is used to hit the playManifest URL directly to fetch the content, there is no need for extra manipulations.

For examples of calls and responses for `getPlaybackContext`, please refer to the Appendix: [Downloading assets from Kaltura - Example of a response to getPlaybackContext](#) | Kaltura Knowledge Center

Uploading vendor REACH Job results to Kaltura

Services input & output

The following table summarizes the assets that REACH vendors download from Kaltura for processing and the assets they upload to Kaltura to fulfil REACH job orders from customers:

	Input (to Download from Kaltura)	Output (to Upload to Kaltura)	Processing and Output Notes

	Input (to Download from Kaltura)	Output (to Upload to Kaltura)	Processing and Output Notes
Caption	Video/Audio	<p>Caption file: SRT/DFXP</p> <p>Related files (ie. transcripts): TXT, DFXP</p>	<p>Captions</p> <p>In the REACH job request, you should receive a <code>catalogItemId</code> in the job details. If <code>serviceFeature=1</code> (for Captions), you can then query the vendor catalog using vendorCatalogItem.get - Kaltura VPaaS API Documentation to retrieve the <code>sourceLanguage</code>.</p> <p>Captions asset are added as CaptionAsset (set <code>captionAsset[accuracy]</code>) using captionAsset.add - Kaltura VPaaS API Documentation</p> <p>Caption files (SRT/DFXP) are uploaded to Kaltura using captionAsset.setContent - Kaltura VPaaS API Documentation</p> <p>The <code>entryVendorTask[outputObjectId]</code> set using entryVendorTask.updateJob - Kaltura VPaaS API Documentation should be set with the value the caption ID created using captionAsset.add - Kaltura VPaaS API Documentation</p> <p>If metadata extraction is enabled on the profiles the vendor needs to generate Tags.</p> <p>Transcript</p> <p>Transcript asset should be added as TranscriptAsset using attachmentAsset.add - Kaltura VPaaS API Documentation</p> <p>Transcript files (TXT and DFXP) are uploaded to Kaltura using attachmentAsset.setContent - Kaltura VPaaS API Documentation</p> <p>The TXT and DFXP ID returned after creating a transcript using attachmentAsset.add - Kaltura VPaaS API Documentation, should be added to the captions file using captionAsset.update - Kaltura VPaaS API Documentation, this ensures the transcript is associated with the captions and the relevant related files.</p>

	Input (to Download from Kaltura)	Output (to Upload to Kaltura)	Processing and Output Notes
Translation	Video/Audio	Caption file: SRT/DFXP Related files (ie. transcripts): TXT, DFXP	<p>In the REACH job request, you should receive a <code>catalogItemId</code> in the job details. If <code>serviceFeature=2</code> (for Translations), you can then query the vendor catalog using vendorCatalogItem.get - Kaltura VPaaS API Documentation to retrieve both the <code>sourceLanguage</code> and the <code>targetLanguage</code>.</p> <p>In the case of translations, Kaltura supports both translations from source captions file and translations from source audio:</p> <ol style="list-style-type: none"> For translations from source captions file ("requireSource": true or when that parameter is not present), the vendor needs to retrieve the captions file in the source language for generating the translation in the target language. This is done using captionAsset.getUrl - Kaltura VPaaS API Documentation. <ol style="list-style-type: none"> The vendor needs to add the caption asset ID of the source caption asset used to generate the translation file in <code>entryVendorTask[outputObjectId]</code> of the job using entryVendorTask.updateJob - Kaltura VPaaS API Documentation For translations from source audio ("requireSource": false), the vendor needs to retrieve the audio flavor. This is done using baseEntry.getPlaybackContext - Kaltura VPaaS API Documentation
Dubbing	Video	Audio flavor	<p>The vendor should extract the original audio and provide that back as the native language track, then also add the relevant dubbed language track.</p> <p>In the REACH job request, you should receive a <code>catalogItemId</code> in the job details. If <code>serviceFeature=7</code> (for dubbing), you can then query the vendor catalog using vendorCatalogItem.get - Kaltura VPaaS API Documentation to retrieve both the <code>sourceLanguage</code> and the <code>targetLanguage</code>.</p> <p>When processing the task and returning the dubbed audio track, the vendor needs to add the original audio track as well. This ensures the Kaltura Player will see multiple language files to choose from and will function properly.</p> <p>Both audio flavors should be added using flavorAsset.add - Kaltura VPaaS API Documentation.</p> <p>The dubbed audio flavor should not be set to default.</p> <p>The original audio / 'clear audio' should be set to default.</p>
Standard Audio Description	Video	Audio Description flavor Audio flavor of the original language	<p>In the REACH job request, you should receive a <code>catalogItemId</code> in the job details. If <code>serviceFeature=4</code> (for audio description), you can then query the vendor catalog using vendorCatalogItem.get - Kaltura VPaaS API Documentation to retrieve further information.</p> <p>Audio Description flavor - includes merged video audio and audio description (the flavor param id is specify on the catalog item)</p> <p>Audio flavor of the original language - clear audio, without the description (the flavor param id is specify on the catalog item)</p>

	Input (to Download from Kaltura)	Output (to Upload to Kaltura)	Processing and Output Notes
Extended Audio Description	Video	VTT file	<p>In the REACH job request, you should receive a <code>catalogItemId</code> in the job details. If <code>serviceFeature=9</code> (for extended audio description), you can then query the vendor catalog using vendorCatalogItem.get - Kaltura VPaaS API Documentation to retrieve further information.</p> <p>When processing the task and returning the VTT captions, the vendor needs to add the VTT file as attachment with a “vtt” file extension using attachmentAsset.add - Kaltura VPaaS API Documentation and add an “AAD” tag. This ensures the Kaltura Player plugin will read the VTT file as an Extended Audio Descriptions source and leverage browser based voice synthesis.</p>
Chaptering	Video	Cue points - chapters	<p>In the REACH job request, you should receive a <code>catalogItemId</code> in the job details. If <code>serviceFeature=5</code> (for chaptering), you can then query the vendor catalog using vendorCatalogItem.get - Kaltura VPaaS API Documentation to retrieve further information.</p> <p>Chaptering assets are added as cuePoint using cuePoint.add - Kaltura VPaaS API Documentation.</p>
Live caption	Scheduled Event	Caption URL and Caption token	<p>In the REACH job request, you should receive a <code>catalogItemId</code> in the job details. If <code>serviceFeature=8</code> (for Live caption), you can then query the vendor catalog using vendorCatalogItem.get - Kaltura VPaaS API Documentation to retrieve further information.</p> <p>For live, you should have a 'scheduleEventId' that comes in the entryVendorTask details, along with the scheduled start and end times for the live event. After provisioning your services for the live session, you should call scheduleEvent.updateLiveFeature() to provide the following data back to Kaltura:</p> <ul style="list-style-type: none"> <code>mediaUrl</code> and <code>mediaKey</code> - an RTMP(S) stream ingest url and key where Kaltura should relay the live stream to you. <code>captionUrl</code> and <code>captionToken</code> - a websocket address and token where Kaltura should connect at the scheduled event time to receive caption data from you over a realtime websocket. <p>The logic is the following:</p> <ol style="list-style-type: none"> 1. User creates a captioning job on REACH 2. The vendor scans their REACH jobs periodically 3. When the vendor finds a new live captions job, they should find a relevant live entry ID on the task object 4. The vendor calls an API and update the scheduled live event with the RTMP and websocket details 5. When the live stream from customer starts, Kaltura restreams it to the RTMP endpoint provided 6. When the scheduled time is due Kaltura connects to the websocket

REACH job details that need to be taken into consideration when processing jobs:

- Service type
- Service Feature
- Turn Around Time
- Expected Finish Time
- Notes

REACH profiles parameters that need to be taken into consideration when processing jobs:

- Default Output Format (refer to the Appendix “Kaltura Enums Related to REACH API”)
- Enable Metadata Extraction
- Enable Audio Tags
- Enable Profanity Removal
- Content Deletion Policy (refer to the Appendix “Kaltura Enums Related to REACH API”)
- Task Processing Region (refer to the Appendix “Kaltura Enums Related to REACH API”)
- Max Characters Per Caption Line
- Flavor Params Ids
- Dictionary

REACH profiles parameters that **do not need** to be taken into consideration when processing jobs:

- Enable Machine Moderation
- Enable Human Moderation
- Label Addition for Machine Service Type
- Label Addition for Human Service Type

Flavor Asset Language - By default media which is uploaded has the `flavorAsset.language` set to undefined because there is no way to know upon upload of the content, what is the language of the media: there is currently no analysis of the media to determine what language it is, therefore the `flavorAsset.language` will be set to “undefined”. The `flavorAsset.language` is usually set by REACH vendors when delivering dubbing orders: when adding a new flavorAsset, vendors also set the `flavorAsset.language` for the dubbed soundtrack. We recommend to process undefined sources, especially in Production. When you process the jobs, you need to consider the `sourceLanguage` as that is explicitly defined by the user who submits the REACH order.

Captions

- To execute API calls against Kaltura you need to first generate a valid KS.
- Processing
 - For each captioning task, the vendor should change the `entryVendorTask[status]` to “Processing” using `entryVendorTask.updateJob` - [Kaltura VPaaS API Documentation](#) before starting handling the task
 - With a REACH job request (`entryVendorTask.get` - [Kaltura VPaaS API Documentation](#)) that specifies the `responseProfile[systemName]=reach_vendor` in the query, you will receive the catalog details and REACH profile information as well.
 - If `serviceFeature=1` (for Captions),
 - You can check the `sourceLanguage`.
 - You should check that the `sourceLanguage` actually matches the soundtrack of the video file submitted for processing, if there is a mismatch the caption job should end in error
 - And of course check for the `outputFormat` value. If there is none then you can use SRT as default.
 - If `serviceType=2` (for Machine),
 - You can choose to download an audio flavor for the purpose of the Machine based translation.
 - You should check the `dictionary` value for words or phrases (delimited by a carriage return) to increase the quality of the deliverable.
 - And you can check for the Speaker Identification, if `enableSpeakerId=1`, vendor needs to add speaker IDs (ie. “[Speaker


1]”) to the caption lines.

- And you can check for the Speaker Change Indication, if `enableSpeakerChangeIndication=1`, vendor needs to add the “>>” characters whenever a new speaker voice pattern is detected.
 - If `serviceType=1` (for Human),
 - You can choose to download a video flavor (for example the source flavor or a compressed flavor) for the Human translator
 - You should ensure the Human translator receives any “notes” sent by the user via the `notes` field in the job details.
 - You should check the `dictionary` value for words or phrases (delimited by a carriage return) to increase the quality of the deliverable.
 - You should ensure the Human translator receives the `expectedFinishTime` and `turnAroundTime` information from the job details.
 - And you can check for the Speaker Identification, if `enableSpeakerId=1`, vendor needs to add speaker IDs (ie. “[Speaker 1]” or “[Interviewer]”) to the caption lines.
 - And you can check for the Speaker Change Indication, if `enableSpeakerChangeIndication=1`, vendor needs to add the “>>” characters whenever a new speaker voice pattern is detected.
 - If metadata extraction is enabled on the profiles `enableMetadataExtraction=true`, the vendor needs to generate Tags.
 - The vendor needs to analyze the generated captions/transcript to extract metadata.
- Asset creation and upload
 - The caption asset is added as **CaptionAsset** (set `captionAsset[accuracy]`) using [captionAsset.add - Kaltura VPaaS API Documentation](#) (retrieve the caption asset ID returned in the response).
 - Setting for the **CaptionAsset**:
 - `captionAsset[format]` should be set to the requested format.
 - `captionAsset[fileExt]` should be set to the relevant format.
 - `captionAsset[accuracy]` should be set to the relevant accuracy.
 - `captionAsset[language]` should be set to the caption language. The list of all languages which can be used is documented on [KalturaCatalogItemLanguage - Kaltura VPaaS API Documentation](#), you should use the values not the key name.
 - `captionAsset[isDefault]` should be set to `1` as it is the source language.
 - `captionAsset[displayOnPlayer]` should be set to `true`
 - `captionAsset[label]` should be set with a value representing the language, for example you can take the same value as for `captionAsset[language]`
 - The caption file (SRT/DXFP) is uploaded to Kaltura using [captionAsset.setContent - Kaltura VPaaS API Documentation](#), the API call offers various file transfer methods, we recommend to use one of the following methods:
 - use `contentResource[objectType]=KalturaAssetResource` to point to an asset you would upload using an [appToken](#) authentication and uploading the produced file to Kaltura using the [chunked upload method](#) and an [uploadToken](#).
 - use `contentResource[objectType]=KalturaUrlResource` and provide a link to the vendor’s deliverable with `contentResource[url]=http://www.vendor.com/deliverable_to_upload` to instruct Kaltura to download the vendor’s deliverable. You can set `contentResource[forceAsyncDownload]=false` to execute the download immediately. Make sure you use `contentResource[urlHeaders][0][value]` and `contentResource[urlHeaders][0][objectType]=KalturaString` to pass authentication tokens to securely transfer data.
 - [OPTIONAL] The transcript asset should be added as **TranscriptAsset** using [attachmentAsset.add - Kaltura VPaaS API Documentation](#) (retrieve the transcript asset ID returned in the response).
 - Setting for the **TranscriptAsset**:
 - `captionAsset[format]` should be set to the requested format.
 - `captionAsset[fileExt]` should be set to the relevant format.

- `captionAsset[accuracy]` should be set to the relevant accuracy.
 - `captionAsset[language]` should be set to the translated language. The list of all languages which can be used is documented on [KalturaCatalogItemLanguage - Kaltura VPaaS API Documentation](#), you should use the values not the key name.
 - `captionAsset[isDefault]` should be set to `0` as it is a translation.
- [OPTIONAL] The transcript file (TXT and DFXP) is uploaded to Kaltura using [attachmentAsset.setContent - Kaltura VPaaS API Documentation](#), the API call offers various file transfer methods, we recommend to use one of the following methods:
 - use `contentResource[objectType]=KalturaAssetResource` to point to an asset you would upload using an `appToken` authentication and uploading the produced file to Kaltura using the [chunked upload method](#) and an `uploadToken`.
 - use `contentResource[objectType]=KalturaUrlResource` and provide a link to the vendor's deliverable with `contentResource[url]=http://www.vendor.com/deliverable_to_upload` to instruct Kaltura to download the vendor's deliverable. You can set `contentResource[forceAsyncDownload]=false` to execute the download immediately. Make sure you use `contentResource[urlHeaders][0][value]` and `contentResource[urlHeaders][0][objectType]=KalturaString` to pass authentication tokens to securely transfer data.
 - [OPTIONAL] After the transcript asset and file are created and uploaded, the vendor needs to update the `associatedTranscriptIds` using [captionAsset.update - Kaltura VPaaS API Documentation](#), with the TXT and DFXP ID returned after creating a transcript using [attachmentAsset.add - Kaltura VPaaS API Documentation](#), this ensures the transcript is associated with the captions and the relevant related files.
 - In the case of translations, the vendor needs to add the caption asset ID of the source caption asset used to generate the translation file in `entryVendorTask[outputObjectId]` of the job using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#)
 - If metadata extraction is enabled on the profiles `enableMetadataExtraction=true`, the vendor needs to create Tags on the entry:
 - The vendor needs to update the entry with a new tag using [media.update - Kaltura VPaaS API Documentation](#) or [baseEntry.update - Kaltura VPaaS API Documentation](#)
 - If there are already existing tag entries, you first need to retrieve the existing tag entries using [media.get - Kaltura VPaaS API Documentation](#) or [baseEntry.get - Kaltura VPaaS API Documentation](#), then append the new tags to the existing tags and then update the tag entries using [media.update - Kaltura VPaaS API Documentation](#) or [baseEntry.update - Kaltura VPaaS API Documentation](#)
 - Supposing you do metadata extraction for Translations, you can then add or update tags using the `mediaEntry[multiLingual_tags]` or `baseEntry[multiLingual_tags]` parameters depending on the call you are using. For example if using media entries you can set French tags with the following parameters:

```
"mediaEntry[multiLingual_tags][0][language]=FR" "mediaEntry[multiLingual_tags][0][value]=francais" "mediaEntry[multiLingual_tags][0][objectType]=KalturaMultiLingualString"
```

- Post processing
 - After the vendor processing is done for the task, the vendor needs to update the `entryVendorTask[outputObjectId]` using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#) with the ID of the asset that contains the output of the task, which in this case is the value of the caption ID created using [captionAsset.add - Kaltura VPaaS API Documentation](#) (the ID is returned in the response)
 - After a successful task, the vendor should change the `entryVendorTask[status]` to "Ready" using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#)

 In case of any unsuccessful task, the vendor should change the `entryVendorTask[status]` to "Error" using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#). When setting this status the `errDescription` field should always be filled with a short description about the failure cause.

For examples of calls and responses for Captions and Translations, please refer to the Appendix:
[Adding captions | Kaltura Knowledge Center](#)

Translations

- To execute API calls against Kaltura you need to first generate a valid KS.
- Processing
 - For each translation task, the vendor should change the `entryVendorTask[status]` to “Processing” using [entryVendorTask.updateJob](#) - [Kaltura VPaaS API Documentation](#) before starting handling the task
 - With a REACH job request ([entryVendorTask.get](#) - [Kaltura VPaaS API Documentation](#)) that specifies the `responseProfile[systemName]=reach_vendor` in the query, you will receive the catalog details and REACH profile information as well.
 - If `serviceFeature=2` (for Translations),
 - If `requireSource=true` or when that parameter is not present, the vendor needs to retrieve the captions file in the source language for generating the translation in the target language. This is done using [captionAsset.getUrl](#) - [Kaltura VPaaS API Documentation](#).
 - You should check whether there is a caption file in the `sourceLanguage` specified. Most importantly, you need to download the caption file of the source language to then generate the translation in the target language. To retrieve the captions file in the source language you need to use [captionAsset.getUrl](#) - [Kaltura VPaaS API Documentation](#) and pass the `captionAssetId` returned in the task
 - You should check the caption accuracy of the source and not translate it if the captions are not at 100% percent accuracy (representing Human caption) and send back an error message.
 - If `requireSource=false`, the vendor needs to retrieve the audio flavor. This is done using [baseEntry.getPlaybackContext](#) - [Kaltura VPaaS API Documentation](#).
 - If `serviceType=2` (for Machine),
 - You can choose to download an audio flavor for the purpose of the Machine based translation.
 - If `serviceType=1` (for Human),
 - You can choose to download a video flavor (for example the source flavor or a compressed flavor) for the Human translator
 - Regardless of the value of `requireSource` :
 - You can check both the `sourceLanguage` and the `targetLanguage`.
 - And of course check for the `outputFormat` value. If there is none then you can use SRT as default
 - If `serviceType=2` (for Machine),
 - You should check the `dictionary` value for words or phrases (delimited by a carriage return) to increase the quality of the deliverable.
 - And you can check for the Speaker Identification, if `enableSpeakerId=1`, vendor needs to add speaker IDs (ie. “[Speaker 1]”) to the caption lines.
 - And you can check for the Speaker Change Indication, if `>enableSpeakerChangeIndication=1`, vendor needs to add the “>>” characters whenever a new speaker voice pattern is detected
 - If `serviceType=1` (for Human),
 - You should ensure the Human translator receives any “notes” sent by the user via the `notes` field in the job details.
 - You should check the `dictionary` value for words or phrases (delimited by a carriage return) to increase the quality of the deliverable.
 - And you can check for the Speaker Identification, if `enableSpeakerId=1`, vendor needs to add speaker IDs (ie. “[Speaker 1]” or “[Interviewer]”) to the caption lines.
 - And you can check for the Speaker Change Indication, if `enableSpeakerChangeIndication=1`, vendor needs to add the “>>” characters whenever a new speaker voice pattern is detected.
 - You should ensure the Human translator receives the `expectedFinishTime` and `turnAroundTime` information from the job details.
 - If metadata extraction is enabled on the profiles `enableMetadataExtraction=true`, the vendor needs to generate Tags.


- The vendor needs to analyze the generated captions/transcript to extract metadata.
- Asset creation and upload
 - The caption asset is added as **CaptionAsset** (set `captionAsset[accuracy]`) using [captionAsset.add - Kaltura VPaaS API Documentation](#) (retrieve the caption asset ID returned in the response).
 - Setting for the **CaptionAsset**:
 - `captionAsset[format]` should be set to the requested format.
 - `captionAsset[fileExt]` should be set to the relevant format.
 - `captionAsset[accuracy]` should be set to the relevant accuracy.
 - `captionAsset[language]` should be set to the caption language. The list of all languages which can be used is documented on [KalturaCatalogItemLanguage - Kaltura VPaaS API Documentation](#), you should use the values not the key name.
 - `captionAsset[isDefault]` should be set to `0` as it is not the source language.
 - `captionAsset[displayOnPlayer]` should be set to `true`
 - `captionAsset[label]` should be set with a value representing the language, for example you can take the same value as for `captionAsset[language]`
 - The caption file (SRT/DXFP) is uploaded to Kaltura using [captionAsset.setContent - Kaltura VPaaS API Documentation](#), the API call offers various file transfer methods, we recommend to use one of the following methods:
 - use `contentResource[objectType]=KalturaAssetResource` to point to an asset you would upload using an [appToken](#) authentication and uploading the produced file to Kaltura using the [chunked upload method](#) and an [uploadToken](#).
 - use `contentResource[objectType]=KalturaUrlResource` and provide a link to the vendor's deliverable with `contentResource[url]=http://www.vendor.com/deliverable_to_upload` to instruct Kaltura to download the vendor's deliverable. You can set `contentResource[forceAsyncDownload]=false` to execute the download immediately. Make sure you use `contentResource[urlHeaders][0][value]` and `contentResource[urlHeaders][0][objectType]=KalturaString` to pass authentication tokens to securely transfer data.
 - [OPTIONAL] The transcript asset should be added as **TranscriptAsset** using [attachmentAsset.add - Kaltura VPaaS API Documentation](#) (retrieve the transcript asset ID returned in the response).
 - Setting for the **TranscriptAsset**:
 - `captionAsset[format]` should be set to the requested format.
 - `captionAsset[fileExt]` should be set to the relevant format.
 - `captionAsset[accuracy]` should be set to the relevant accuracy.
 - `captionAsset[language]` should be set to the translated language. The list of all languages which can be used is documented on [KalturaCatalogItemLanguage - Kaltura VPaaS API Documentation](#), you should use the values not the key name.
 - `captionAsset[isDefault]` should be set to `0` as it is a translation.
 - [OPTIONAL] The transcript file (TXT and DFXP) is uploaded to Kaltura using [attachmentAsset.setContent - Kaltura VPaaS API Documentation](#), the API call offers various file transfer methods, we recommend to use one of the following methods:
 - use `contentResource[objectType]=KalturaAssetResource` to point to an asset you would upload using an [appToken](#) authentication and uploading the produced file to Kaltura using the [chunked upload method](#) and an [uploadToken](#).
 - use `contentResource[objectType]=KalturaUrlResource` and provide a link to the vendor's deliverable with `contentResource[url]=http://www.vendor.com/deliverable_to_upload` to instruct Kaltura to download the vendor's deliverable. You can set `contentResource[forceAsyncDownload]=false` to execute the download immediately. Make sure you use `contentResource[urlHeaders][0][value]` and `contentResource[urlHeaders][0][objectType]=KalturaString` to pass authentication tokens to securely transfer data.
 - [OPTIONAL] After the transcript asset and file are created and uploaded, the vendor needs to update the `associatedTranscriptIds` using [captionAsset.update - Kaltura VPaaS API Documentation](#), with the TXT and DFXP ID returned after creating a transcript using [attachmentAsset.add - Kaltura VPaaS API Documentation](#), this ensures the transcript is associated with the captions and the relevant related files.

- In the case of translations, the vendor needs to add the caption asset ID of the source caption asset used to generate the translation file in `entryVendorTask[outputObjectId]` of the job using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#)
- If metadata extraction is enabled on the profiles `enableMetadataExtraction=true`, the vendor needs to create Tags on the entry:
 - The vendor needs to update the entry with a new tag using [media.update - Kaltura VPaaS API Documentation](#) or [baseEntry.update - Kaltura VPaaS API Documentation](#)
 - If there are already existing tag entries, you first need to retrieve the existing tag entries using [media.get - Kaltura VPaaS API Documentation](#) or [baseEntry.get - Kaltura VPaaS API Documentation](#), then append the new tags to the existing tags and then update the tag entries using [media.update - Kaltura VPaaS API Documentation](#) or [baseEntry.update - Kaltura VPaaS API Documentation](#)
 - Supposing you do metadata extraction for Translations, you can then add or update tags using the `mediaEntry[multiLingual_tags]` or `baseEntry[multiLingual_tags]` parameters depending on the call your are using. For example if using media entries you can set French tags with the following parameters:

```
"mediaEntry[multiLingual_tags][0][language]=FR" "mediaEntry[multiLingual_tags][0][value]=francais" "mediaEntry[multiLingual_tags][0][objectType]=KalturaMultiLingualString"
```



- Post processing
 - After the vendor processing is done for the task, the vendor needs to update the `entryVendorTask[outputObjectId]` using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#) with the ID of the asset that contains the output of the task, which in this case is the value of the caption ID created using [captionAsset.add - Kaltura VPaaS API Documentation](#) (the ID is returned in the response)
 - After a successful task, the vendor should change the `entryVendorTask[status]` to "Ready" using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#)

 In case of any unsuccessful task, the vendor should change the `entryVendorTask[status]` to "Error" using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#). When setting this status the `errDescription` field should always be filled with a short description about the failure cause.

For examples of calls and responses for Translations, please refer to the Appendix:
[Adding translations | Kaltura Knowledge Center](#)

Audio tracks for dubbing and standard audio descriptions

- To execute API calls against Kaltura you need to first generate a valid KS.
- Processing
 - For each captioning task, the vendor should change the `entryVendorTask[status]` to "Processing" using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#) before starting handling the task
 - With a REACH job request ([entryVendorTask.get - Kaltura VPaaS API Documentation](#)) that specifies the `responseProfile[systemName]=reach_vendor` in the query, you will receive the catalog details and REACH profile information as well.
 - If `serviceFeature=7` (for dubbing),
 - You can then check both the `sourceLanguage` and the `targetLanguage`.
 - You probably should choose to download a video flavor (for example the source flavor or a compressed flavor) for the Human translator to ensure good timing of the produced dubbing audio track, but the vendor could optionally choose to download an audio flavor.
 - Make sure you retrieve the values of both `flavorParamsId` and `clearAudioFlavorParamsId` in the REACH vendor catalog item object, those indicate the audio flavors to use for both audio tracks the Vendor will produce and upload to Kaltura (`flavorParamsId` will be to be used for the dubbed track and `clearAudioFlavorParamsId` will be to be used for the source track).
 - If `serviceFeature=4` (for audio description), you can then check the `sourceLanguage`.

- You must choose to download a video flavor (for example the source flavor or a compressed flavor) for the Human translator
- You should ensure the Human translator receives the information from the job details in order to ensure he uses the natural pauses of the video to add audio descriptions.
- Because `serviceType=1` (for Human),
 - You should ensure the Human translator receives the `expectedFinishTime` and `turnAroundTime` information from the job details.
- Asset creation and upload
 - When processing the task and returning the dubbed audio track or the audio description track, the vendor needs to add the original audio track as well. This ensures the Kaltura Player will see multiple language files to choose from and will function properly.
 - Both in case of dubbing and standard audio description, both **audio flavor assets** should be added using [flavorAsset.add - Kaltura VPaaS API Documentation](#) (retrieve the flavor asset ID returned in the response).
 - Setting for the **dubbed audio flavor or the audio description track:**
 - `flavorAsset[isDefault]` should be set to `0` as it is not the source language
 - `flavorAsset[language]` should be set to the dubbed language or the audio description language. The list of all languages which can be used is documented on [KalturaCatalogItemLanguage - Kaltura VPaaS API Documentation](#), you should use the values not the key name.
 - `flavorAsset[isOriginal]` should be set to `0` as it is not the source language
 - `flavorAsset[fileExt]` should be set to the relevant format
 - For **dubbed audio flavor (only):**
 - The `flavorParamsId` should be set to the value of the `flavorParamsId` parameter retrieved in the REACH vendor catalog item object of the job ([entryVendorTask.get - Kaltura VPaaS API Documentation](#))
 - Setting for the **original audio flavor:**
 - `flavorAsset[isDefault]` should be set to `1` as it is the source language
 - `flavorAsset[language]` should be set to the source language. The list of all languages which can be used is documented on [KalturaCatalogItemLanguage - Kaltura VPaaS API Documentation](#), you should use the values not the key name.
 - `flavorAsset[isOriginal]` should be set to `1` as it is the source language
 - The `flavorParamsId` should be set to the value of the `clearAudioFlavorParamsId` parameter retrieved in the REACH vendor catalog item object of the job ([entryVendorTask.get - Kaltura VPaaS API Documentation](#))
 - `flavorAsset[fileExt]` should be set to the relevant format
 - Both **audio flavor files** should be uploaded using [flavorAsset.setContent - Kaltura VPaaS API Documentation](#), the API call offers various file transfer methods, we recommend to use one of the following methods:
 - use `contentResource[objectType]=KalturaAssetResource` to point to an asset you would upload using an [appToken](#) authentication and uploading the produced file to Kaltura using the [chunked upload method](#) and an [uploadToken](#).
 - use `contentResource[objectType]=KalturaUrlResource` and provide a link to the vendor's deliverable with `contentResource[url]=http://www.vendor.com/deliverable_to_upload` to instruct Kaltura to download the vendor's deliverable. You can set `contentResource[forceAsyncDownload]=false` to execute the download immediately. Make sure you use `contentResource[urlHeaders][0][value]` and `contentResource[urlHeaders][0][objectType]=KalturaString` to pass authentication tokens to securely transfer data.
- Post processing
 - After the vendor processing is done for the task, the vendor needs to update the `entryVendorTask[outputObjectId]` using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#) with the ID of the asset that contains the output of the task, which in this case is the value of the audio flavor ID created using [flavorAsset.add - Kaltura VPaaS API Documentation](#) (the ID is returned in the response)
 - After a successful task, the vendor should change the `entryVendorTask[status]` to "Ready" using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#)

In case of any unsuccessful task, the vendor should change the `entryVendorTask[status]` to “Error” using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#). When setting this status the `errDescription` field should always be filled with a short description about the failure cause.

For examples of calls and responses for Audio tracks, please refer to the [Appendix: Adding audio flavors for dubbing and standard audio descriptions | Kaltura Knowledge Center](#)

Extended audio descriptions

- To execute API calls against Kaltura you need to first generate a valid KS.
- Processing
 - For each extended audio description task, the vendor should change the `entryVendorTask[status]` to “Processing” using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#) before starting handling the task
 - With a REACH job request ([entryVendorTask.get - Kaltura VPaaS API Documentation](#)) that specifies the `responseProfile[systemName]=reach_vendor` in the query, you will receive the catalog details and REACH profile information as well.
 - With `serviceFeature=9` (for extended audio description), you can then check the `sourceLanguage` and the `outputFormat`.
 - You should ensure the Human translator receives the information from the job details in order to ensure he uses adds time codes for his audio descriptions and delivers a VTT captions file.
 - Because `serviceType=1` (for Human),
 - You should ensure the Human translator receives the `expectedFinishTime` and `turnAroundTime` information from the job details.
- Asset creation and upload
 - When processing the task and returning the VTT captions, the vendor needs to:
 - add the VTT file as an `attachmentAsset` the attachment using [attachmentAsset.add - Kaltura VPaaS API Documentation](#) and add the following information:
 - `"attachmentAsset[tags]=AAD"` to ensure the Player will read this file using speech synthesis
 - `"attachmentAsset[fileExt]=vtt"`
 - `"attachmentAsset[title]"` to set a Human readable name which will be displayed in the Player, for example “Extended Audio Description”
 - upload the VTT file using [attachmentAsset.setContent - Kaltura VPaaS API Documentation](#), the API call offers various file transfer methods, we recommend to use one of the following methods:
 - use `contentResource[objectType]=KalturaAssetResource` to point to an asset you would upload using an [appToken](#) authentication and uploading the produced file to Kaltura using the [chunked upload method](#) and an [uploadToken](#).
 - use `contentResource[objectType]=KalturaUriResource` and provide a link to the vendor’s deliverable with `contentResource[url]=http://www.vendor.com/deliverable_to_upload` to instruct Kaltura to download the vendor’s deliverable. You can set `contentResource[forceAsyncDownload]=false` to execute the download immediately. Make sure you use `contentResource[urlHeaders][0][value]` and `contentResource[urlHeaders][0][objectType]=KalturaString` to pass authentication tokens to securely transfer data.
 - Post processing
 - After the vendor processing is done for the task, the vendor needs to update the `entryVendorTask[outputObjectId]` using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#) with the ID of the asset that contains the output of the task, which in this case is the value of the audio flavor ID created using [attachmentAsset.add - Kaltura VPaaS API Documentation](#) (the ID is returned in the response)
 - After a successful task, the vendor should change the `entryVendorTask[status]` to “Ready” using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#)

In case of any unsuccessful task, the vendor should change the `entryVendorTask[status]` to “Error” using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#). When setting this status the `errDescription` field should always be filled with a short description about the failure cause.

For examples of calls and responses for Extended Audio Descriptions, please refer to the Appendix: [Adding extended audio descriptions | Kaltura Knowledge Center](#)

Chaptering

- To execute API calls against Kaltura you need to first generate a valid KS.
- Processing
 - For each chaptering task, the vendor should change the `entryVendorTask[status]` to “Processing” using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#) before starting handling the task
 - With a REACH job request ([entryVendorTask.get - Kaltura VPaaS API Documentation](#)) that specifies the `responseProfile[systemName]=reach_vendor` in the query, you will receive the catalog details and REACH profile information as well.
 - With `serviceFeature=5` (for chaptering), you can then check the `sourceLanguage`.
 - If `serviceType=1` (for Human),
 - You should ensure the Human translator receives the `expectedFinishTime` and `turnAroundTime` information from the job details.
- Asset creation and upload
 - When processing the task and returning the cuepoints, the vendor needs to create cuepoints. There are two options:
 1. Create cuepoints one by one using [cuePoint.add - Kaltura VPaaS API Documentation](#), in such case the vendor can build a reusable error handling logic to be used for each call
 2. Bulk create cuepoints using [cuePoint.addFromBulk - Kaltura VPaaS API Documentation](#), in such case the vendor needs to process the error of the bulk operation and proceed depending on the error details
- Post processing
 - After the vendor processing is done for the task, the vendor needs to update the `entryVendorTask[outputObjectId]` using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#) with the ID of the asset that contains the output of the task, which in this case is the value of the audio flavor ID created using [attachmentAsset.add - Kaltura VPaaS API Documentation](#) (the ID is returned in the response)
 - After a successful task, the vendor should change the `entryVendorTask[status]` to “Ready” using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#)

Live captions

- To execute API calls against Kaltura you need to first generate a valid KS.
- Processing
 - For each chaptering task, the vendor should change the `entryVendorTask[status]` to “Processing” using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#) before starting handling the task
 - With a REACH job request ([entryVendorTask.get - Kaltura VPaaS API Documentation](#)) that specifies the `responseProfile[systemName]=reach_vendor` in the query, you will receive the catalog details and REACH profile information as well.
 - With `serviceFeature=8` (for Live caption), you can then check the `sourceLanguage`.
 - If `serviceType=1` (for Human),
 - You should ensure the Human translator receives the scheduled `startDate` and `endDate` information from the job details.

- Asset generation
 - Call [scheduleEvent.updateLiveFeature - Kaltura VPaaS API Documentation](#) to update the Kaltura event with the following fields so that we can send you the stream data, and retrieve the needed caption data:
 - `scheduleEventId` : this is the `scheduleEventId` that you should have received in the `entryVendorTask` request
 - `featureName` : for live captions, the `featureName` string should follow the format `LiveCaptionFeature-reach-<taskId>` where `taskId` is the id of the `entryVendorTask` you are fulfilling
 - `liveFeature` : use the “KalturaLiveCaptionFeature” object with the following fields:
 - `captionToken` : this should be a security token used when accessing the websocket that you will be outputting the caption data stream to
 - `captionUrl` : this is the url (Vendor hosted) of the websocket that you will be outputting the caption data stream to
 - `mediaKey` : this is the stream key/name for the RTMP(S) stream where we will send you a relay of the media stream
 - `mediaUrl` : this is the RTMP(S) stream URL (Vendor hosted) where we will send you a relay of the media stream
 - Update the `entryVendorTask` status to SCHEDULED using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#).

Optional - The vendor may also put their task id in the `externalTaskId` field

The `entryVendorTask` remains in the scheduled status until the start of the event. The client can still abort the task in this state and the task status will change to ABORTED (the client will only receive a refund if they do so before the `minimalRefundTime` that is configured on the catalog item).

- At the time of the live session:
 - Once the event starts and the vendor starts receiving content and sending captions they need to update the `entryVendorTask` status to PROCESSING using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#).
 - Kaltura will relay the media stream to your provided `mediaUrl/mediaKey` for you to process the stream and output the caption feed back on the `captionUrl/captionToken` websocket that you provide.
 - Kaltura connects to the vendor’s websocket by the URL given to us by the vendor over the `liveFeature`. Authentication of the websocket can happen in 2 layers (up to the vendor) :
 - First we authenticate by pre-given token set in the general Kaltura secret with the vendor (regardless of this specific event). The auth URL for this step is also given by the vendor.
 - The second layer is per-stream authentication. The vendor can set the per-session token on the `liveFeature` and define the way to use that (will be implemented in the specific vendor adaptor) or can add the auth data over the caption websocket URL provided by the vendor.
 - Kaltura will handle ingesting the caption data and transforming to WebVTT format to inject into the player along with the HLS stream that viewers will consume.
[See Websocket response schema for an example of a JSON caption data payload to be returned by the websocket.](#)
- Post processing
 - After the vendor processing is done for the task, the vendor needs to update the `entryVendorTask[outputObjectId]` using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#) with the ID of the asset that contains the output of the task, which in this case is the value of the audio flavor ID created using [attachmentAsset.add - Kaltura VPaaS API Documentation](#) (the ID is returned in the response)
 - After a successful task, the vendor should change the `entryVendorTask[status]` to “Ready” using [entryVendorTask.updateJob - Kaltura VPaaS API Documentation](#)

Currently if the customer starts streaming before the REACH vendor adds the details for the task, they will not receive captions for their stream at all.

For examples of calls and responses for Live captions, please refer to the Appendix: [Adding Live captions | Kaltura Knowledge Center](#)

Job Prioritization

The vendor should have a queueing mechanism that gives equal priority to all jobs, regardless of the customer that initiated the request.

]

[template("cat-subscribe")]
