Working with the Kaltura Reach API

Last Modified on 04/24/2026 7:59 pm IDT

This article is designated for administrators.

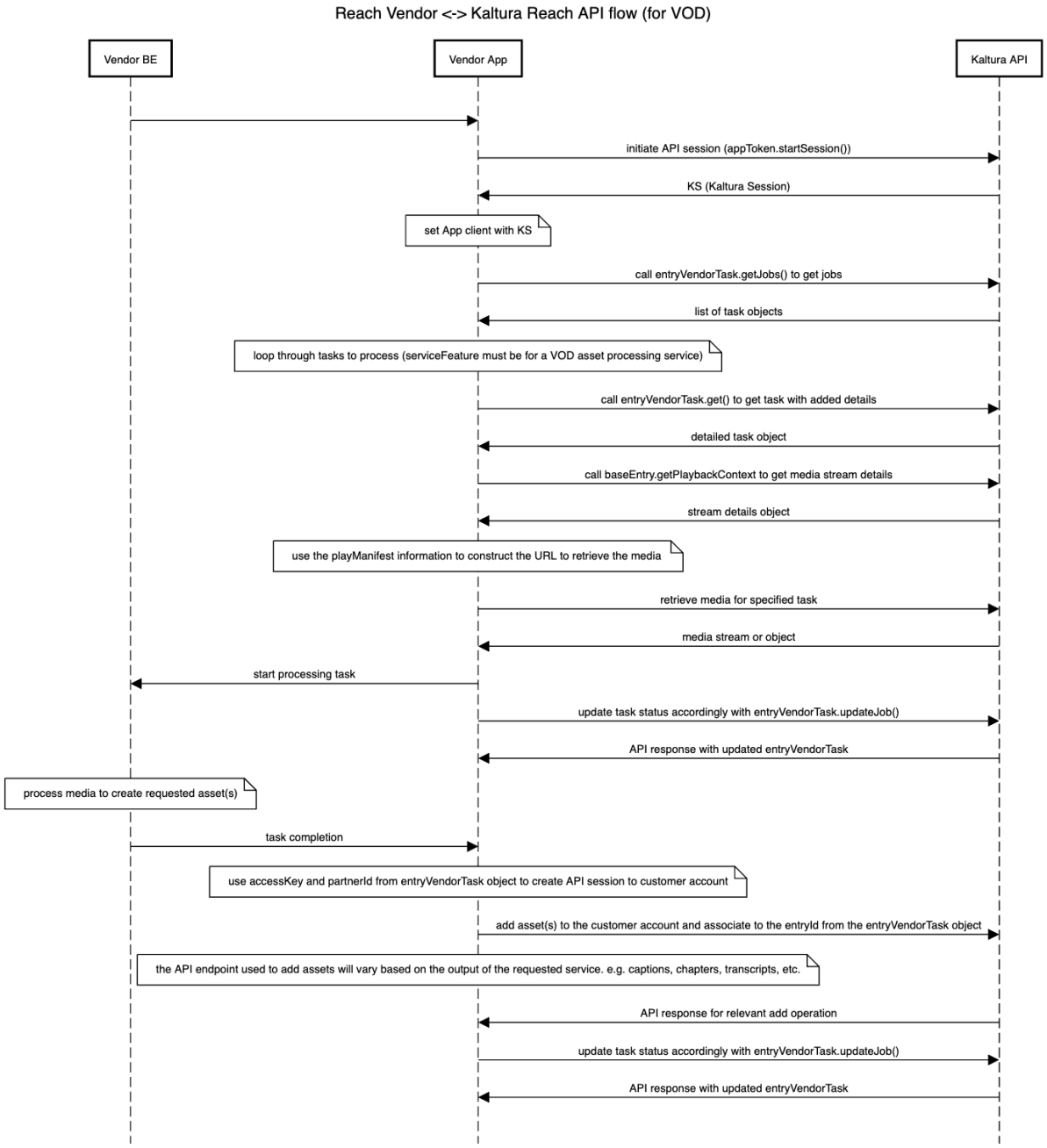
About

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

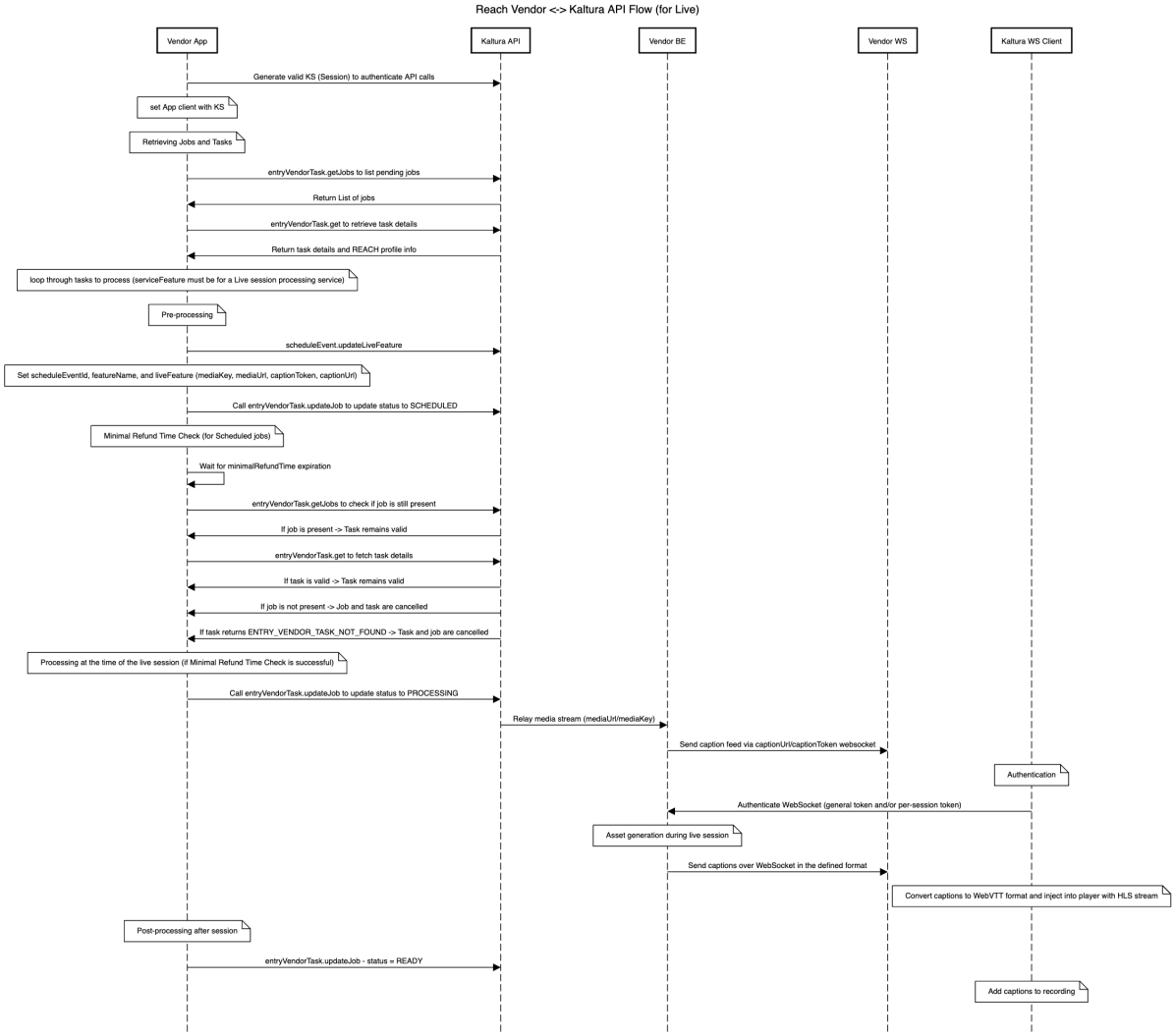
Vendors get access to special APIs (entryVendorTask - Kaltura VPaaS API Documentation) which allows them to periodically pull new Reach tasks (each Task will contain metadata required for the vendor to process the request), process them and finally, update the customer's Kaltura account with the resulting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



Reach Vendor API - Basic Flow (for Live sessions)



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 non-task-specific API calls, the `clientTag` should be set as '*<default clientTag>_vendorName-vendorPartnerId*'. (*<default ClientTag>* consists of the client library's programming language and the library's 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 non-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".

- **Service URLs** - If you use a client library, make sure you specify the correct `serviceUrl` as per indications in the "Reach V2 - Vendor Data privacy and Content deletion" documentation. For example when using the Python client library:
    - when creating a `KalturaConfiguration`, you can send in a `serviceUrl`: `config = KalturaConfiguration()`

      ```
      config.serviceUrl = "https://www.kaltura.com/"
      client = KalturaClient(config)
      ```

    - when not sending a `serviceUrl`, it will default to what is set in the KalturaConfiguration which is using HTTP: KalturaGeneratedAPIClientsPython/KalturaClient/Base.py at dcac7d15f9654f4a2a81971d712d4ed04341f112 · kaltura/KalturaGeneratedAPIClientsPython
    - when not sending a `serviceUrl`, the client library will return a SERVICE_ACCESS_CONTROL_RESTRICTED error.
    - As good practice you should always set the `serviceUrl`, especially if you start supporting the various Kaltura SaaS regions which have different domain names.

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 expiration time, the default is 24 hours.
    - The KS needs 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 gathering 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 as 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 its `entryVendorTask[status]` to “Processing” using entryVendorTask.updateJob - Kaltura VPaaS API Documentation before beginning to handle 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 will 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 an 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 its `status` to “Processing” before starting to handle 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 an 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 response 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), including the Reach vendor profile information:

```
{
  "objects":[
    {
      "id":"15",
      "partnerId":110,
      "vendorPartnerId":108,
      "createdAt":1521639688,
      "entryId":"0_o2z3458w",
      "status":1,
      "reachProfileId":1,
      "catalogItemId":1,
      "accessKey":"OTNmNDE5Zjk2MDQxMTZkZWRhZjg1NTkzNTkxZDY5NDRjM2Y1ZmNlZnwxMTA7MTEwOzE1MjE3MjYwODg7MDsxNTIxNjM5Njg4LjQyMTM7O2VkaXQ6MF9vMnozNDU4dyxzZXRyb2x
      "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

An access key validity depends on the catalog item's TaT (Turn Around Time): the value is calculated to be the max value between 7 days and twice the value set in the catalog item's TaT.

In the case that the vendor processing took longer than expected and the access key provided on the task 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.

> The extendAccessKey action must be called with the Vendor KS. Generally, you can call it multiple times but that will not be needed as each extension gives you twice the turn-around time defined on the catalog item.

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 vendors 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 follow it strictly 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 where 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 where 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 still want a captioning or transcript. In such a 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 it should 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=OWRhZTUyNTI3ZGQ4YjMzYWM3ZWJiY2RhYTAwODNmNzY2ZjY1ZWM1N3wxMTk7MTE5OzE1NDcxMDk2MDc7MjsxNTQ3MDIzMjA3LjYwMjQ7ZWxsYS5saWRpY2hAa2FsdHVyYS5jb207KixkaXNhYr
-d "entryId=0_tpavjp5t" \
-d "contextDataParams[streamerType]=applehttp" \
-d "contextDataParams[ks]=OWRhZTUyNTI3ZGQ4YjMzYWM3ZWJiY2RhYTAwODNmNzY2ZjY1ZWM1N3wxMTk7MTE5OzE1NDcxMDk2MDc7MjsxNTQ3MDIzMjA3LjYwMjQ7ZWxsYS5saWRpY2hAa2FsdHVy
-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 the case that 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.

If 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 information such as **width**, **height**, **bitrate** 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 the 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. If 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 the 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
```

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

In the 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

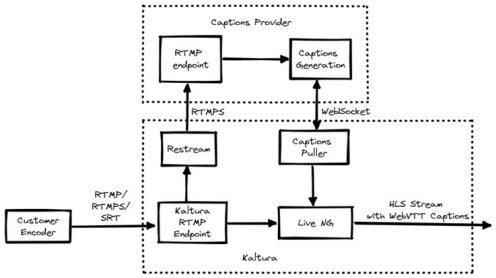
Services input & output

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**
Caption	Video/Audio	**Caption file**: SRT/DFXP **Related files (ie. transcripts)**: TXT, DFXP	**Captions** In the Reach job request, you should receive a `catalogItemId` in the job details. If `serviceFeature=1` (for Captions), you can then query the vendor catalog using vendorCatalogItem.get - Kaltura VPaaS API Documentation to retrieve the `sourceLanguage`. Captions asset are added as **CaptionAsset** (set `captionAsset[accuracy]`) using captionAsset.add - Kaltura VPaaS API Documentation Caption files (SRT/DXFP) are uploaded to Kaltura using captionAsset.setContent - Kaltura VPaaS API Documentation The `entryVendorTask[outputObjectId]` 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 If metadata extraction is enabled on the profiles the vendor needs to generate Tags. **Transcript** Transcript asset should be added as **TranscriptAsset** using attachmentAsset.add - Kaltura VPaaS API Documentation Transcript files (TXT and DFXP) are uploaded to Kaltura using attachmentAsset.setContent - Kaltura VPaaS API Documentation

	Input (to Download from Kaltura)	Output (to Upload to Kaltura)	Processing and Output Notes
			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.
Translation	Video/Audio	**Caption file**: SRT/DFXP **Related files (ie. transcripts)**: TXT, DFXP	In the Reach job request, you should receive a `catalogItemId` in the job details. If `serviceFeature=2` (for Translations), you can then query the vendor catalog using vendorCatalogItem.get - Kaltura VPaaS API Documentation to retrieve both the `sourceLanguage` and the `targetLanguage`. In the case of translations, Kaltura supports both translations from source captions file and translations from source audio: 1. 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. 2. 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 The `entryVendorTask[outputObjectId]` 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
Dubbing	Video	Audio flavor	The vendor should extract the original audio and provide that back as the native language track, then also add the relevant dubbed language track. In the Reach job request, you should receive a `catalogItemId` in the job details. If `serviceFeature=7` (for dubbing), you can then query the vendor catalog using vendorCatalogItem.get - Kaltura VPaaS API Documentation to retrieve both the `sourceLanguage` and the `targetLanguage`. 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. The `entryVendorTask[outputObjectId]` set using entryVendorTask.updateJob - Kaltura VPaaS API Documentation should be set with the value the flavor asset ID created using https://developer.kaltura.com/api-docs/service/flavorAsset/action/add Both **audio flavors** should be added using flavorAsset.add - Kaltura VPaaS API Documentation. The **dubbed audio flavor** should not be set to default. The **original audio / 'clear audio'** should be set to default.
			In the Reach job request, you should receive a `catalogItemId` in the job details. If `serviceFeature=4` (for audio description), you can then query the vendor catalog using vendorCatalogItem.get - Kaltura VPaaS API Documentation to retrieve further information.

	Input (to Download from Kaltura)	Output (to Upload to Kaltura)	Processing and Output Notes
		Audio Description flavor	**Audio flavor of the original language** - includes merged video audio and audio description (the flavor param id is specify on the catalog item)
Standard Audio Description	Video	Audio flavor of the original language	**Audio flavor of the original language** - clear audio, without the description (the flavor param id is specify on the catalog item) The `entryVendorTask[outputObjectId]` set using entryVendorTask.updateJob - Kaltura VPaaS API Documentation should be set with the value the flavor asset ID created using https://developer.kaltura.com/api-docs/service/flavorAsset/action/add
Extended Audio Description	Video	VTT file	In the Reach job request, you should receive a `catalogItemId` in the job details. If `serviceFeature=9` (for extended audio description), you can then query the vendor catalog using vendorCatalogItem.get - Kaltura VPaaS API Documentation to retrieve further information. When processing the task and returning the VTT captions, • historically, the vendor needed 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 ensured the Kaltura Player plugin will read the VTT file as an Extended Audio Descriptions source and leverage browser based voice synthesis. **This method is now deprecated**. • currently, it is recommended that the vendor adds the VTT file as a caption with a `captionAsset[usage]` attribute set to 1, This ensures the Kaltura Player plugin will read the VTT file as an Extended Audio Descriptions source and leverage browser based voice synthesis. The vendor also needs to set the `captionAsset[displayOnPlayer]` attribute to false as it is not necessary to display EAD caption files as part of captions. There are additional attributes to set and those are documented in the relevant section of this documentation. The VTT file added for EAD should have the timestamps based on the original video timestamps. The `entryVendorTask[outputObjectId]` 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
Chaptering	Video	Cue points - chapters	In the Reach job request, you should receive a `catalogItemId` in the job details. If `serviceFeature=5` (for chaptering), you can then query the vendor catalog using vendorCatalogItem.get - Kaltura VPaaS API Documentation to retrieve further information. Chaptering assets are added as **cuePoint** using cuePoint.add - Kaltura VPaaS API Documentation . The `entryVendorTask[outputObjectId]` set using entryVendorTask.updateJob - Kaltura VPaaS API Documentation should be set with the value the cuepoint ID created using https://developer.kaltura.com/api-docs/service/cuePoint/action/add

	Input (to Download from Kaltura)	Output (to Upload to Kaltura)	Processing and Output Notes
Live caption	Scheduled Event	Caption URL and Caption token	In the Reach job request, you should receive a catalogItemId in the job details. If `serviceFeature=8` (for Live caption), you can then query the vendor catalog using vendorCatalogItem.get - Kaltura VPaaS API Documentation to retrieve further information. 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: • `mediaUrl` and `mediaKey` - an RTMP(S) stream ingest url and key where Kaltura should relay the live stream to you. • `captionUrl` and `captionToken` - a websocket address and token where Kaltura should connect at the scheduled event time to receive caption data from you over a realtime websocket. The logic is the following: 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 scheduledEventId 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 
Sign language	Video	Video	In the REACH job request, you should receive a catalogItemId in the job details. If `serviceFeature=19` (for Sign Language), you can then query the vendor catalog using vendorCatalogItem.get - Kaltura VPaaS API Documentation to retrieve further information. The main steps to deliver the Sign language video to Kaltura are: 1. Create draft Video (`baseEntry.add` / `media.add`). 2. Set the original video (step 1) as the 'parentEntryId' of the draft video ('child'; step 4). 3. Add admin tags a. to indicate it is a sign language entry ('signLang') b. language (e.g., 'signLangASL')

	Input (to Download from Kaltura)	Output (to Upload to Kaltura)	Processing and Output Notes
			c. and vendor name. 4. Add the child entry in the response under the [outputObjectId] field.
			The entryVendorTask[outputObjectId] set using entryVendorTask.updateJob - Kaltura VPaaS API Documentation should be set with the value the entry ID of the child entry created using https://developer.kaltura.com/api-docs/service/media/action/add or https://developer.kaltura.com/api-docs/service/baseEntry/action/add

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 `flavorAssset.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.
- We recommend you then check the asset status to verify if the content was downloaded by Kaltura, if not you should error. In this case as it is a captionAsset, you should use captionAsset.get - Kaltura VPaaS API Documentation and proceed when status is `READY [2]` and error if status is `ERROR [-1]`.

- [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.
    - We recommend you then check the asset status to verify if the content was downloaded by Kaltura, if not you should error. In this case as it is an attachmentAsset, you should use attachmentAsset.get - Kaltura VPaaS API Documentation and proceed when status is `READY [2]` and error if status is `ERROR [-1]`.
- [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 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.update - Kaltura VPaaS API Documentation (the ID is returned in the response)
- 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)
- [RECOMMENDED] The vendor may also put their task id in the externalTaskId field of the task
- 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.

ℹ 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 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.
      - We recommend you then check the asset status to verify if the content was downloaded by Kaltura, if not you should error. In this case as it is a captionAsset, you should use captionAsset.get - Kaltura VPaaS API Documentation and proceed when status is `READY [2]` and error if status is `ERROR [-1]`.
  - [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.
    - We recommend you then check the asset status to verify if the content was downloaded by Kaltura, if not you should error. In this case as it is an attachmentAsset, you should use attachmentAsset.get - Kaltura VPaaS API Documentation and proceed when status is `READY [2]` and error if status is `ERROR [-1]`.
- [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)
  - [RECOMMENDED] The vendor may also put their task id in the externalTaskId field of the task
  - 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 description

- 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.
      - 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).
      - 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.
  - The generated audio tracks need to be MP4 and should use AAC-LC Codec with 64Kbps or 128Kbps bitrate.
  - 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[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)
  - The `flavorAsset[label]` should be set with a value representing the language, for example you can take the same value as for `flavorAsset[language]` i.e. “English”
- For **audio description**:
  - 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), for example the value “200” when integrating with the Kaltura US SaaS region. This will ensure that the “audio_only,alt_audio,audio_description” tags are added to the `flavorAsset[tags]` parameter.
  - The `flavorAsset[label]` should be set with a value representing the language with an “Audio description” suffix, for example you can take the same value as for `flavorAsset[language]` i.e. “English” and add “ - Audio description” (note the leading space).

- 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.
  - 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.
    - We recommend you then check the asset status to verify if the content was downloaded by Kaltura, if not you should error. In this case as it is a flavorAsset, you should use flavorAsset.get - Kaltura VPaaS API Documentation and proceed when status is `READY [2]` and error if status is `ERROR [-1]`.
- Some customers like to have a caption file which is aligned with the new spoken audio (dub). It is possible to also upload a caption file but it is important that it is set with the appropriate attributes:
  - be set with a proper “accuracy” value which naturally will be lower than 99%
  - be set with a clear label, for example “Japanese (Dub translation)”
  - language be set, for example “Japanese”
  - display on player be set to TRUE
  - default be set to FALSE

    The above will ensure that if there are multiple translated captions for one same language, they can be easily identified by the end-user.

- 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)
- [RECOMMENDED] The vendor may also put their task id in the externalTaskId field of the task
- 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 description

- 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
  - When processing the task and returning the VTT captions, the vendor needs to create a VTT file where the timestamps are based on the original video timestamps. The Kaltura player takes into account both the start and the end timestamp in the EAD VTT in the following way: if the user skips (forward/backward) to a position that is between the start and end timestamp, the EAD text will be read out loud by the player; the Reach vendor should ensure to indicate both the start and end timestamps based on when the event on the screen start and ends.

    For example: if the video contains a "woman with yellow shirt sitting on a couch" between 00:05 and 00:10, the beginning timestamp in the VTT should be 00:05 and the end timestamp should be 00:10.
- Asset upload (historical / deprecated method):
  - add the VTT file as an `attachmentAsset`, the attachment is added 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]=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.
  - We recommend you then check the asset status to verify if the content was downloaded by Kaltura, if not you should error. In this case as it is an attachmentAsset, you should use attachmentAsset.get - Kaltura VPaaS API Documentation and proceed when status is `READY [2]` and error if status is `ERROR [-1]`.

- Asset upload (current / recommended method):
  - add the VTT file as a `captionAsset`, the caption is added using captionAsset.add - Kaltura VPaaS API Documentation and the add the following information:
    - `"captionAsset[label]=English%20Extended%20Audio%20Description"` although this won't be displayed to the end-user, especially if `captionAsset[displayOnPlayer]=false` is set, it is recommended to add a descriptive label such as "Language + Extended Audio Description"
    - `"captionAsset[language]=English"` allows distinguishing different EAD files in different languages
    - `"captionAsset[isDefault]=0"` to ensure the EAD is not read by default by the Kaltura Player
    - `"captionAsset[displayOnPlayer]=false"` to ensure the EAD is not displayed as part of the normal caption files in the Kaltura player
    - `"captionAsset[format]=3"` to explicitly indicate the file format as VTT
    - `"captionAsset[objectType]=KalturaCaptionAsset"` to indicate the file is a caption file
    - `"captionAsset[usage]=1"` to explicitly indicate to the Kaltura player to treat and display the contents of the VTT file as Extended Audio Descriptions
  - upload the VTT file 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.
      - We recommend you then check the asset status to verify if the content was downloaded by Kaltura, if not you should error. In this case as it is a captionAsset, you should use captionAsset.get - Kaltura VPaaS API Documentation and proceed when status is `READY [2]` and error if status is `ERROR [-1]`.

- 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 cap ID created using captionAsset.add - Kaltura VPaaS API Documentation (the ID is returned in the response)
  - [RECOMMENDED] The vendor may also put their task id in the externalTaskId field of the task
  - 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 cuePoint.add - Kaltura VPaaS API Documentation (the ID is returned in the response)
    - [RECOMMENDED] The vendor may also put their task id in the externalTaskId field of the task
    - After a successful task, the vendor should change the `entryVendorTask[status]` to “Ready” using entryVendorTask.updateJob - Kaltura VPaaS API Documentation

Live captions

- Pre requisites
  - A user creates a new entryVendorTask (assuming the task was created before the minimal order time) that includes these fields in the task data:
    - Start/End time – when they want the live captioning. Epoch time that is checked against the durationLimit.
    - scheduledEventId- id of the related scheduled event object. The object represents a planned live stream (Such as a town hall meeting, lecture, class, etc.
  - To execute API calls against Kaltura you need to first generate a valid KS.
- Retrieving tasks
  - The vendor should uses entryVendorTask.getJobs - Kaltura VPaaS API Documentation action to list pending jobs
  - The vendor should uses entryVendorTask.get - Kaltura VPaaS API Documentation action to list pending tasks
  - 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.

- Pre-processing
    - The vendor updates the Kaltura scheduled event so that Kaltura can send you the stream data, and retrieve the needed caption data
    - The vendor adds the relevant data using the scheduleEvent.updateLiveFeature - Kaltura VPaaS API Documentation action with the following fields :
        - `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 is the id of the entryVendorTask you are fulfilling
        - `liveFeature` : use the “KalturaLiveCaptionFeature” object with the following fields:
            - `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
            - `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
    - The vendor updates the `entryVendorTask[status]` to SCHEDULED using entryVendorTask.updateJob - Kaltura VPaaS API Documentation to ensure the customer is aware that the vendor is ready
    - [RECOMMENDED] The vendor may also put their task id in the externalTaskId field of the task
    - 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).
    - Before an event starts, the vendor can verify all SCHEDULED jobs. Any job which is still in SCHEDULED state can be executed, any job which isn't any longer present was cancelled. It is possible to filter with `filter[statusEqual]=9`

- Processing (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.
    - For each live caption task, the vendor should change the `entryVendorTask[status]` to PROCESSING using entryVendorTask.updateJob - Kaltura VPaaS API Documentation before starting handling the task
    - 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.
    - When live streaming begin, 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) :
        1. 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
        2. The second layer is per-stream authentication. The vendor can set the per-session token on the live feature and define the way to use that (will be implement in the specific vendor adaptor) or can add the auth data over the caption WebSocket URL provided.

- Asset generation (During the Live session)
    - The vendor provides captions text over the WebSocket in the defined format (please refer to examples in Appendix)
    - 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.

- Post processing (Live session ended)

- Once the event ends and the content input is disconnected, 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 Cente

Sign language

- To execute API calls against Kaltura you need to first generate a valid KS.
- Processing
  - For each sign language 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=19` (for sign language), you can then check both `sourceLanguage` and `targetLanguage`.
- Asset creation and upload
  - When processing the task and returning the Sign Language video, the vendor needs to create a child entry:
    1. create a a draft/template video using https://developer.kaltura.com/api-docs/service/media/action/add or https://developer.kaltura.com/api-docs/service/baseEntry/action/add using the following Name convention:
       - Parent: '*NAME*'
       - Child: '*NAME child + signLang[language]*' where [language] is ASL or BSL
       - `curl -X POST https://www.kaltura.com/api_v3/service/media/action/add \`
         `-d "ks=$KALTURA_SESSION" \`
         `-d "entry[objectType]=KalturaMediaEntry" \`
         `-d "entry[name]=MyVideo child signLangASL"`
    2. upload the video file using https://developer.kaltura.com/api-docs/service/media/action/addContent or https://developer.kaltura.com/api-docs/service/baseEntry/action/addContent , 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.
         - We recommend you then check the asset status to verify if the content was downloaded by Kaltura, if not you should error. In this case as it is a captionAsset, you should use captionAsset.get - Kaltura VPaaS API Documentation and proceed when status is `READY [2]` and error if status is `ERROR_CONVERTING [-1]` or `ERROR_IMPORTING [-2]`.
    3. Set the "Child" entry (Sign language video) and bind it to the "Parent" entry using https://developer.kaltura.com/api-docs/service/media/action/update or https://developer.kaltura.com/api-docs/service/baseEntry/action/update and the following attributes:

```
curl -X POST https://www.kaltura.com/api_v3/service/media/action/update \
   -d "ks=$KALTURA_SESSION" \
   -d "entryId=childentry" \
   -d "mediaEntry[objectType]=KalturaMediaEntry" \
   -d "mediaEntry[parentEntryId]=parententry"
   -d "mediaEntry[adminTags]=signLangASL"
```

4. Add admin tags with language (see example above),
    a. The vendor needs to use the following Name convention: signLang[language] - e.g. *signLangASL*
    b. The vendor needs to update the entry with new admin tags using media.update - Kaltura VPaaS API Documentation or baseEntry.update - Kaltura VPaaS API Documentation
        a. it is mandatory to add a '*signLang[language]*' tag where [language] is ASL or BSL
        b. it is mandatory to add a 'singLang' tag to indicate the child entry is to be used for sign language.
        c. and we recommend adding a "*vendorname*" tag where vendorname is representing the vendor name
        d. admin tags are comma separated
    c. 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

- 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 entry ID of the child entry created using https://developer.kaltura.com/api-docs/service/media/action/add or https://developer.kaltura.com/api-docs/service/baseEntry/action/add (the ID is returned in the response)
    - [RECOMMENDED] The vendor may also put their task id in the externalTaskId field of the task
    - After a successful task, the vendor should change the `entryVendorTask[status]` to "Ready" using entryVendorTask.updateJob - Kaltura VPaaS API Documentation

Job prioritization

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