OOYALA INGESTION GUIDE

CONTENTS

COPYRIGHT NOTICE

ABOUT INGESTING CONTENT

Ooyala allows you to upload your videos into Backlot using any of the following options:

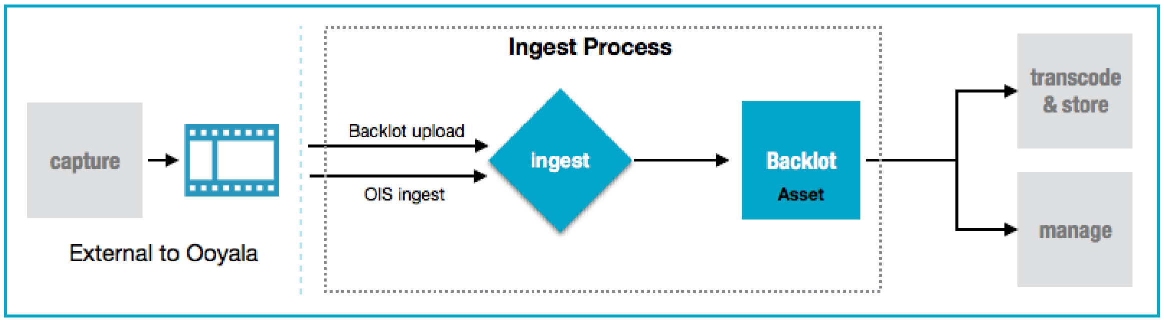
- Backlot directly using the Backlot UI or the Backlot API
- Ooyala Ingestion Services (OIS) using FTP, Aspera, or MRSS

Ingested content can include the video file, thumbnail files, closed caption files, and metadata that makes it easier to find the video in Backlot. Once ingested, your video becomes an asset in Backlot that you can manage, publish, and monetize.

Note: This topic applies to both OIS v2.5 and OIS v1. Certain sections, where indicated, apply to just one version. OIS v1 has been disabled. Customers using OIS v1 should switch to OIS v2.5.

INGEST RUN-TIME WORKFLOW

The following figure shows what occurs how a video asset is ingested into Backlot.



After someone (external to Ooyala) has captured a video and saved it it as a video file encoded in a raw format, you can upload or ingest the video file into your Backlot account, along with other related files (thumbnails, closed caption files, and metadata). Once stored in Backlot as an asset, your video can be transcoded to multiple stream formats and managed as an asset. For an overview of these processes, see *Introducing the Online Video Platform*.

WHAT'S DIFFERENT IN OIS V2.5

Note: Ooyala is migrating customers from OIS v1 to OIS v2.5. Once complete, OIS v1 will be deprecated and eventually disabled. If you have any questions, contact your Ooyala representative.

The Ooyala Ingestion Service (OIS) v2.5 represents a major step forward for uploading videos and metadata into Backlot. OIS v2.5 makes it easier for users to ingest files into Backlot using FTP, Aspera, or MRSS. This section summarizes the differences between OIS v2.5 and OIS v1.

Note: Changes are with FTP, Aspera, and MRSS ingest. Ingesting with the Backlot UI and Backlot API has not changed in this version.

New Ingestion Endpoints for OIS v2.5

Log in using your Backlot upload-only user account credentials (email and password). You can use these endpoints for either FTP or Aspera clients.

Region	Endpoint
All	transfer.ooyala.com
US	transfer-us.ooyala.com
EU	transfer-eu.ooyala.com

Region	Endpoint
APAC	transfer-apac.ooyala.com

The main endpoint automatically redirects to the region closest to you. You can also access the region-specific endpoints directly.

Enablement for Manifest-based Ingestion

For OIS v2.5, the process of enabling manifest-based ingestion has been streamlined, resulting in a faster turnaround for enablement requests. Your request automatically includes support for both XML and CSV manifests (you no longer request just one or the other). Basic ingestion is already enabled by default.

Upload-only user accounts

To access the OIS v2.5 endpoints, you use upload-only users associated with your Ooyala account. In the Backlot GUI (ACCOUNT > User Management), you can add a user with Upload Only permissions to your account (see *Managing Users*). You can then use the email and password associated with this user to log into the new ingestion endpoint at transfer.ooyala.com when using either an FTP or Aspera client. By creating upload-only users, you no longer need to submit a request to Ooyala Support to obtain special credentials for FTP or Aspera.

Ingesting via FTP or Aspera

For OIS v2.5, files are deleted right after they are processed. Submitted files that do not go through the entire ingestion workflow (for example, awaiting the upload of a manifest file) are deleted from the server after 7 days.

OIS v2.5 Uses FTPS (FTP Over SS)

You need to use an FTPS client when ingesting with FTP.

Updating Remote Assets Using MRSS

OIS v2.5 adds the ability to update remote assets in Backlot (using MRSS feeds) with new metadata, thumbnails, and closed caption files.

Creating Mobile Assets using CSV

With OIS v2.5, you can create remote assets using CSV ingestion. This adds a third option to existing support for remote asset creation using XML and MRSS.

WHAT YOU CAN INGEST

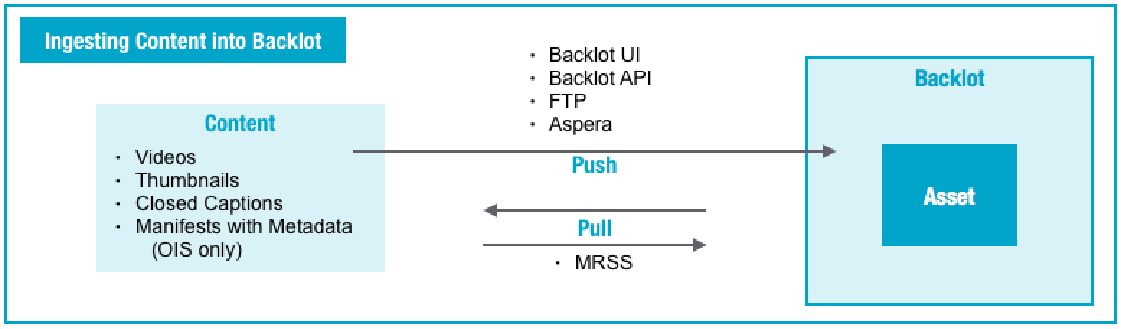
You can ingest the following types of files.

File Type	Description	Requirements
video files	Any video files with a supported format. Includes audio-only files (video files with only an audio soundtrack) and VR 360 videos.	*Supported Ingest Formats*
thumbnail files	Low resolution images that are generated at consistent intervals during playback as users scrub through a video. Used for the Preview image.	*Supported Ingest Formats*
closed captions files	A subtitle or closed caption file contains both the text of what is said in the video and time codes for when each line of text should be displayed.	*Ingesting Closed Caption Files* on page 29

File Type	Description	Requirements
manifest files	In addition to uploading video files, you can specify extra metadata (including the video title, description, flight times, labels, and custom metadata), as well as thumbnail and closed caption files. You specify these options in a manifest file (in CSV or XML format) that you also upload.	*Manifest File Formats* on page 17
remote assets	Video that is stored remotely outside of Backlot but managed within Ooyala and delivered through an Ooyala player.	*Remote Assets*

WAYS TO INGEST

Ooyala provides you with the following ingest options.



Backlot Direct Uploads

Ingestion Mechanism	For more information
Backlot UI	*Ingesting with the Backlot UI* on page 10
Backlot API	Specifically the `/v2/assets` routes. See *Ingesting with the Backlot API* on page 10.

Ooyala Ingest Services (OIS)

Use OIS (with or without manifests) to ingest content via the following transport mechanisms:

Transport Mechanism	For more information
FTP	*Ingesting with FTP / FTPS* on page 11
Aspera	*Ingesting with Aspera* on page 13
MRSS	*Ingesting from a Remotely Hosted MRSS Feed* on page 15

BASIC AND MANIFEST-BASED INGEST

If you are using FTP or Aspera to ingest videos, there are two approaches: basic and manifest-based ingest. MRSS requires a manifest-based ingest.

Note: For OIS v2.5, if you want to use both basic and manifest-based ingestion, you need to create separate, uniquely-named, upload-only user accounts for each (for example, `user1_basic@example.com` and `user1_manifest@example.com`).

Basic Ingest (Videos Files Only)

With basic ingest, you upload only video, thumbnail, and closed caption files. Basic ingest is automatically enabled for all accounts. The only requirement is that you add an upload-only user to your account in Backlot, configure the user password, and then use the upload-only user's email address and password to log into the ingest endpoint.

Once ingested, you can use Backlot to configure asset settings and manage the asset, including:

Content	Backlot UI	For more information
video title and description	Manage > Details subtab > Title, Description	*Managing Video Details*
thumbnails	Manage > Details subtab > Preview Image	*Managing the Preview Image*
flight times	Manage > Publishing Rules subtab > Flight Times	*Managing Publishing Rules for Channel Sets (Deprecated)*
closed captions	Manage > Details subtab	*Uploading a Closed Caption File in Backlot*
custom metadata	Manage > Custom Metadata subtab	*Managing Custom Metadata*
MRSS	Publish -> External Publishing	*Syndication with Source MRSS*

Manifest-based Ingest

With manifest-based ingest, in addition to uploading video files, you can specify extra metadata (including the video title, description, flight times, labels, and custom metadata), as well as thumbnail and closed caption files. You specify these options in a *manifest file* (in CSV or XML format) that you also upload. See *Manifest File Formats* on page 17.

Manifest-based ingestion is disabled by default. To enable this feature, you must:

1. Create an upload-only user in Backlot and configure the user password (see *Managing Users*).
2. Create a support request to Ooyala to enable manifest-based ingestion. Include the email address associated with the upload-only user for your account.

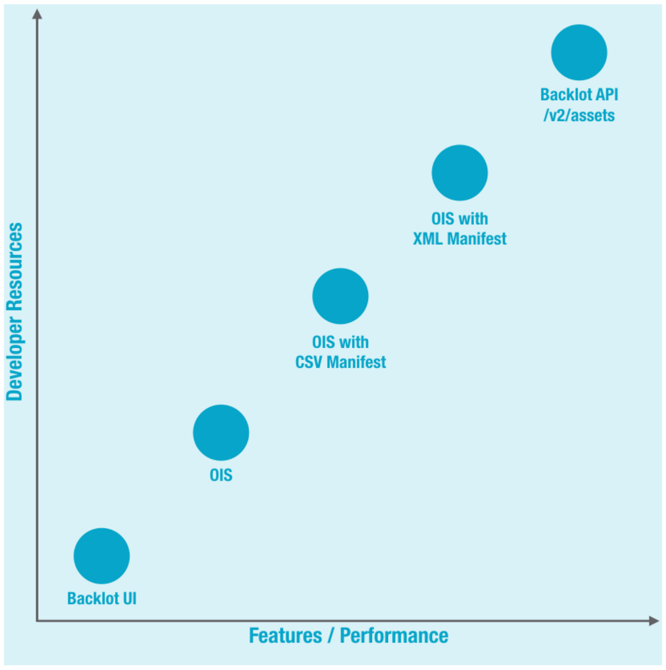
Upon approval, your account will be enabled for both XML and CSV manifest formats.

Dynamic Manifests

In addition to the static CSV and XML manifest file formats, you can use filters to apply dynamic manifest criteria to assets. See *Dynamic Manifests* on page 26.

DECIDING THE BEST INGEST OPTION

The figure below shows your Ooyala ingest options, plotted by required development time and by features/ performance.



Refer to the following table to help you decide which option is best for you.

If...	then use...
you have just a few few files to upload and have no programming resources available	Backlot UI. After uploading a video, you can use the Backlot UI to manually associate it with ad sets, specify metadata and configure other features.
you have many files to upload (bulk ingestion)	FTP or Aspera, with or without a manifest file.
you have many files to upload (bulk ingestion) and want to: • include thumbnails (preview images) with them, or • associate custom metadata with them	FTP or Aspera with a manifest file in CSV format.
you want Ooyala to pull files from your content management system (CMS) (that is, to use MRSS)	MRSS with a manifest file in XML format.
you have programming resources	Backlot API directly.

Note: If you intend to use manifest files with FTP or Aspera, or you intend to use MRSS, contact your Customer Success Manager or Technical Support.

OIS INGEST LOG

In the Backlot UI, upload-only users can view a read-only report of ingest jobs that they have submitted via OIS (see *Viewing the Ingest Log in Backlot*). For programmatic access to the OIS ingest log, see the *Ingest Log REST API*.

IN-REGION INGESTION AND TRANSCODING

Leveraging Azure's Media Service, Ooyala Ingestion Service and Ooyala Transcoding Service support ingestion and transcoding of video assets performed in your local region. This reduces transcoding time and keeps content geographically contained. In-region ingestion is supported for files uploaded using the Backlot API, FTP, or Aspera.

To see whether your account is eligible for in-region ingestion and transcoding, contact your Ooyala account representative.

Limitations:

- Not supported for processing profiles that include asset files in Flash format
- Not supported for processing profiles that mix clear and DRM assets
- Does not support multi-DRM configuration for the same muxing format
- Does not support thumbnail
- Does not support closed captions

CONTENT REPLACEMENT

You cannot use content replacement with the FTP, Aspera, or MRSS mechanisms. Attempting to do so will result in duplicate assets.

Instead, use the `/v2/assets` API request, with the `/replacement` qualifier for each asset. See the *Ingesting with the Backlot API* on page 10.

WAYS TO INGEST CONTENT

You can ingest videos and other content using the following options.

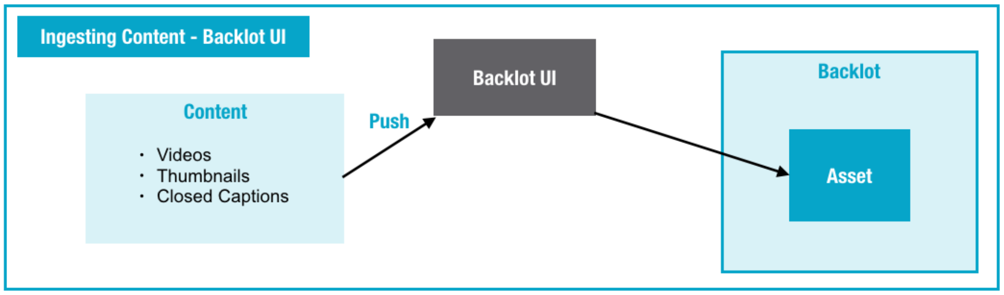
UPLOADING TO BACKLOT

You can upload videos and other content directly into Backlot.

Ingesting with the Backlot UI

Use the Backlot UI to manually upload content into Backlot.

You can use the Backlot UI to manually upload and manage videos, thumbnails, and closed caption files.



Once ingested, you can use Backlot to configure asset settings and manage the asset, including:

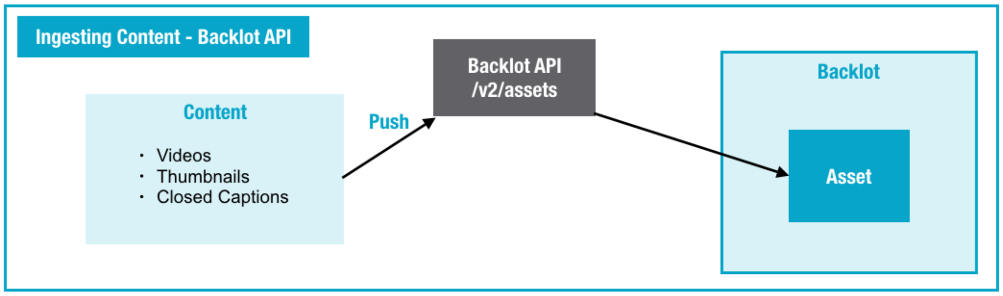
Content	Backlot UI	For more information
video title and description	Manage > Details subtab > Title, Description	*Managing Video Details*
thumbnails	Manage > Details subtab > Preview Image	*Managing the Preview Image*
flight times	Manage > Publishing Rules subtab > Flight Times	*Managing Publishing Rules for Channel Sets (Deprecated)*
closed captions	Manage > Details subtab	*Uploading a Closed Caption File in Backlot*
custom metadata	Manage > Custom Metadata subtab	*Managing Custom Metadata*
MRSS	Publish -> External Publishing	*Syndication with Source MRSS*

The Backlot UI is a manual upload process. To upload large numbers of videos, there are automated and more efficient ways (see *About Ingesting Content* on page 4.). If you want to use the Backlot UI for occasional uploading, see *Uploading a Video*.

Ingesting with the Backlot API

Use the Backlot API to programmatically upload content into Backlot. This approach enables you to integrate your content management system (CMS) or workflows directly with the Backlot platform.

You can use the Backlot API to programmatically upload and manage videos, thumbnails, and closed caption files.



The Backlot API provides a high level of integration and customization. It requires development time and resources. If you want to ingest content using the Backlot API, see

- *Video and Audio Assets* (for videos and thumbnails)
- *Working with Closed Captions*

The primary call used to upload files (called "assets") is the `/v2/assets` route.

Note: Do not add a video with a null `external_id`, that is, an `external_id` with no value (`""`) or a value of `"null"`. Such null external IDs cannot be searched for later.

OOYALA INGESTION SERVICE (OIS)

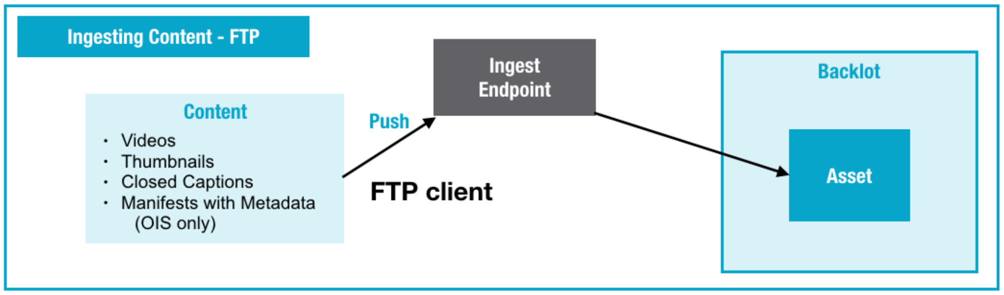
You can use the following OIS options to ingest videos and other content.

Ingesting with FTP / FTPS

You can use FTPS (recommended) or FTP (deprecated) to upload videos, thumbnails, closed captions, and manifest files to Backlot.

Note: This topic applies to both OIS v2.5 and OIS v1. Certain sections, where indicated, apply to just one version. OIS v1 has been disabled. Customers using OIS v1 should switch to OIS v2.5.

OVERVIEW OF FTP INGEST



ABOUT FTP / FTPS

FTP is a standard network protocol for transferring files from one computer to another. For OIS v2.5, FTPS (FTP over SSL) is required. FTP is used for OIS v1, which is deprecated.

Most operating systems support FTP natively, and you can choose from many FTP client applications. You can automate FTP uploads by writing scripts, have users bulk upload videos through a Windows, Mac, or Linux application, or have users bulk upload videos through a browser (e.g., FireFTP).

Note: You can use any FTP client to upload videos to Backlot.

ACCOUNT POLICY

Your account is limited to a maximum of 100GB on the ingestion server at any time. If you expect to upload more content, you can request a temporary increase from your Customer Success Manager or Technical Support.

GETTING INGEST CREDENTIALS

You need to obtain ingest credentials in order to upload files to Ooyala via FTP.

Setting Up an Upload-only User in Backlot (OIS v2.5)

Note: If you want to use both basic and manifest-based ingestion, you need to create separate, uniquely-named, upload-only user accounts for each (for example, `user1_basic@example.com` and `user1_manifest@example.com`).

1. In the Backlot GUI (Account > User Management), add a user with Upload Only permissions to your account (see *Managing Users*).
2. Configure the password for this account.
3. Use the email and password associated with this user to log into the ingestion endpoint (see *Ooyala Ingest Server Endpoints* on page 16).

Note: If you want to use manifest-based ingest, contact Ooyala to enable this functionality for your account.

SETTING VR 360 FILE METADATA

If you are uploading a VR 360 video file, first label the file as VR 360 and tell which type of VR 360 it is, monoscopic or stereoscopic. Use file metadata to transmit this information.

You can set VR 360 file metadata in the manifest file (see *Manifest File Formats* on page 17).

If you are not using a manifest file, indicate the file type by setting file metadata with the `ffmpeg` command-line utility. For example, suppose you are going to upload a Matroska (.mkv) file. With this type of file, you can create a custom metadata tag named `vr360type` and assign it one of the following values: `mono` or `stereo_lr`. For example:

```
ffmpeg -i inputFile.mkv -c copy -metadata "vr360type=mono" outputFile.mkv
```

Some file formats do not support custom metadata. For example, with MP4 files, `ffmpeg` allows only predefined metadata tags. As a workaround, use the general-purpose `description` field with predefined values to indicate the VR 360 file type. The value of `description` must include the text `VR 360` and one of the following: `mono` or `stereo_lr`. For example:

```
ffmpeg -i inputFile.mp4 -c copy -metadata "description"="VR 360 Type is
 mono ..."  outputFile.mp4
```

For more information about VR 360, see *VR 360 Videos*.

INGESTING WITH AN FTP / FTPS CLIENT

You can upload a video via FTP / FTPS either with or without a manifest.

Note: All upload files must comply with the ingest requirements described in *Supported Ingest Formats*

Uploading Your File(s) via FTPS (OIS v2.5)

Note: OIS v2.5 requires an FTPS client.

1. **1.** Prepare the file(s) you want to upload.
2. **2.** Launch your FTPS client program.
3. **3.** Log into the ingest endpoint (see *Ooyala Ingest Server Endpoints* on page 16) using the credentials (email and password) of your Backlot upload-only user account.
4. **4.** Upload the file(s) you want to ingest (videos, thumbnails, closed caption files, and so on). If you are using a manifest file, be sure to upload all of the files that are referenced in the manifest file.
5. **5.** Ooyala suggests that you upload the manifest file last.

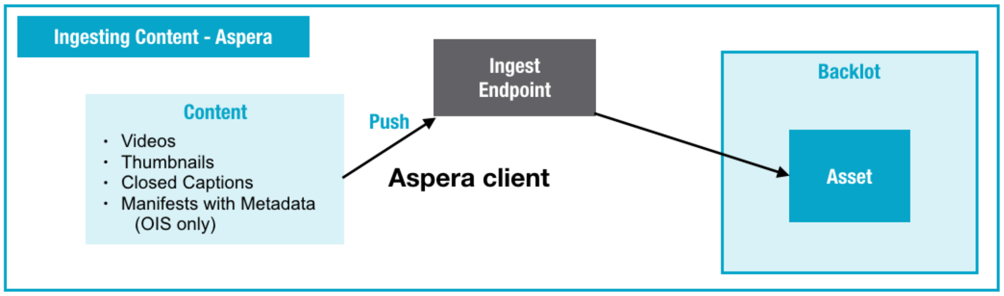
Files are deleted right after they are all processed. Submitted files that do not go through the entire ingestion workflow (for example, awaiting the upload of the manifest file) are deleted from the server after 7 days.

Ingesting with Aspera

You can use Aspera file transfer software to upload videos, thumbnails, closed captions and manifest files to Backlot.

Note: This topic applies to both OIS v2.5 and OIS v1. Certain sections, where indicated, apply to just one version. OIS v1 has been disabled. Customers using OIS v1 should switch to OIS v2.5.

OVERVIEW OF ASPERA INGEST



ABOUT ASPERA

Aspera provides a high-performance file transfer client for uploading videos to Backlot. Aspera uses encryption while transferring data.

ACCOUNT POLICY

Your account is limited to a maximum of 100GB on the ingestion server at any time. If you expect to upload more content, you can request a temporary increase from your Customer Success Manager or Technical Support.

GETTING INGEST CREDENTIALS

You need to obtain ingest credentials in order to upload files to Ooyala via Aspera.

Setting Up an Upload-only User in Backlot (OIS v2.5)

Note: If you want to use both basic and manifest-based ingestion, you need to create separate, uniquely-named, upload-only user accounts for each (for example, `user1_basic@example.com` and `user1_manifest@example.com`).

1. **1.** In the Backlot GUI (Account > User Management), add a user with Upload Only permissions to your account (see *Managing Users*).
2. **2.** Configure the password for this account.

3. Use the email and password associated with this user to log into the ingestion endpoint (see *Ooyala Ingest Server Endpoints* on page 16).

Note: If you want to use manifest-based ingest, contact Ooyala to enable this functionality for your account.

DOWNLOADING THE ASPERA CLIENT

Aspera is available as a free browser plug-in or as a licensed desktop client. The Aspera browser plugin will be offered for download the first time a user connects to the ingest endpoint and logs in successfully. You can also download the software separately.

- To download the free browser plugin, go to *Aspera downloads page* and download the Aspera Connect plugin.
- To download the desktop client, go to *aspera client*.

SETTING VR 360 FILE METADATA

If you are uploading a VR 360 video file, first label the file as VR 360 and tell which type of VR 360 it is, monoscopic or stereoscopic. Use file metadata to transmit this information.

You can set VR 360 file metadata in the manifest file (see *Manifest File Formats* on page 17).

If you are not using a manifest file, indicate the file type by setting file metadata with the `ffmpeg` command-line utility. For example, suppose you are going to upload a Matroska (.mkv) file. With this type of file, you can create a custom metadata tag named `vr360type` and assign it one of the following values: `mono` or `stereo_lr`. For example:

```
ffmpeg -i inputFile.mkv -c copy -metadata "vr360type=mono" outputFile.mkv
```

Some file formats do not support custom metadata. For example, with MP4 files, `ffmpeg` allows only predefined metadata tags. As a workaround, use the general-purpose `description` field with predefined values to indicate the VR 360 file type. The value of `description` must include the text `VR 360` and one of the following: `mono` or `stereo_lr`. For example:

```
ffmpeg -i inputFile.mp4 -c copy -metadata "description"="VR 360 Type is
 mono ..."  outputFile.mp4
```

For more information about VR 360, see *VR 360 Videos*.

INGESTING WITH THE ASPERA CLIENT

You can upload a video via the Aspera client either with or without a manifest.

Note: All upload files must comply with the ingest requirements described in *Supported Ingest Formats*

Uploading Your File(s) via the Aspera Client (OIS v2.5)

1. Prepare the file(s) you want to upload.
2. In a browser, enter the endpoint transfer.ooyala.com. This endpoint will work from all locations. You can also use a specific endpoint for your region. See *Ooyala Ingest Server Endpoints* on page 16.
3. Log into the ingest endpoint using the credentials (email and password) of your Backlot user account.
4. If prompted, open the Aspera app. (If no prompt appears, the Aspera app is already open.)
5. Upload the file(s) you want to ingest (videos, thumbnails, closed caption files, and so on). If you are using a manifest file, be sure to upload all of the files that are referenced in the manifest file.
6. Ooyala suggests that you upload the manifest file last.

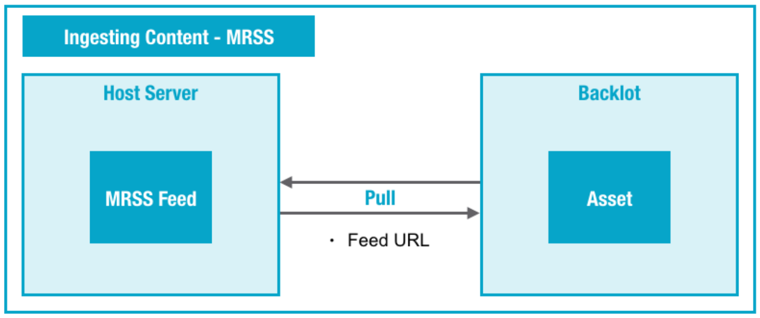
Files are deleted immediately after they are all processed. Submitted files that do not go through the entire ingestion workflow (for example, awaiting the upload of the manifest file) are deleted from the server after 7 days.

Ingesting from a Remotely Hosted MRSS Feed

You can use Media RSS (MRSS) to have Ooyala pull your content from your own system from an MRSS feed that you publish. Backlot pulls the metadata from the MRSS feed and either creates a remote asset or downloads (from where they are hosted) and processes the videos and any associated files (thumbnails, closed caption files, etc.).

Note: This topic applies to both OIS v2.5 and OIS v1. Certain sections, where indicated, apply to just one version. OIS v1 has been disabled. Customers using OIS v1 should switch to OIS v2.5.

OVERVIEW OF MRSS INGEST



ABOUT MRSS

Media RSS (MRSS) is an extension of RSS that allows you to syndicate multimedia files in RSS feeds. For technical details, see the *Media RSS Specification*.

SETTING UP MRSS

Contact your Ooyala representative and indicate that you want to use MRSS. For each video you want to ingest, you must provide Ooyala with valid content URLs. URLs can include `http://` or `https://`.

USING MRSS

To use MRSS, you host metadata and/or videos and other files (thumbnails, closed captions) on a web server, and create an XML manifest file that references that content.

Using MRSS to Pull Metadata Only

1. Create an XML manifest file as described in *XML Manifest File* on page 19. You must use the `ooyala:remoteasset` tag for metadata-only ingestion.
2. Host the XML manifest file on one of the MRSS locations you provided. Make sure that the filename and path matches exactly one of the paths you provided to Ooyala.

Using MRSS to Pull Videos, Thumbnails, and Closed Caption Files

1. Host one or more files (videos, thumbnails, and closed caption files) on a web server.
2. Create an XML manifest file (see *XML Manifest File* on page 19). Make sure that the entries in the file reference the files on your server. In particular, the value of the `<media:content>` element's `href` attribute *must* be an HTTP URL to the file on your server, as in the following example:

```
<media:content url="http://mysite.com/upload/lacrosse_70.mov" />
```

3. Host the XML manifest file on one of the MRSS locations you provided. Make sure that the filename and path exactly matches one of the paths you provided to Ooyala.

   Note: Unfortunately, the concept of "client logging" does not apply to MRSS feed-based processing. The best way to verify that your feed is processing successfully is to check for the appearance of your assets, either with the Backlot UI or the Backlot API `/v2/assets` request.

After the videos and any other associated files are processed, to pull more files, you simply host another MRSS file in the agreed-upon location.

VALIDATING MRSS FEEDS

You can use the following feed validator to validate the MRSS feeds you create: Ooyala *MRSS Feed Validator*.

VALIDATING CLOSED CAPTIONS

Use the Ooyala *Closed Captions Validator* to check your DFXP/TTML documents for your convenience.

Note: We provide Ooyala-specific DFXP support. Even if you follow the w3c document definition for DFXP/TTML, it may not be supported. If you run into any issues with your DFXP/TTML files, contact your Ooyala support representative.

Ooyala Ingest Server Endpoints

Note: This topic applies to both OIS v2.5 and OIS v1. Certain sections, where indicated, apply to just one version. OIS v1 has been disabled. Customers using OIS v1 should switch to OIS v2.5.

Use the following endpoint servers to upload content to Ooyala via FTP and Aspera. You must use the appropriate credentials to log into these servers.

OISV2.5 INGEST SERVER ENDPOINTS

Log in using your Backlot upload-only user account credentials (email and password). You can use these endpoints for either FTP or Aspera clients.

Region	Endpoint
All	transfer.ooyala.com
US	transfer-us.ooyala.com
EU	transfer-eu.ooyala.com
APAC	transfer-apac.ooyala.com

The main endpoint automatically redirects to the region closest to you. You can also access the region-specific endpoints directly.

OIS V1 SERVER ENDPOINTS (DEPRECATED)

Note: OIS v1 has been disabled. Any customers using OIS v1 must switch to OIS v2.5.

Aspera Endpoint

Use the following endpoint for Aspera client uploads: `aspera.upload.ooyala.com`

FTP Endpoints

Use the following endpoints for FTP client uploads.

Region	Endpoint
US	`ftp.upload.ooyala.com`
UK	`uk.ftp.upload.ooyala.com`
JP	`jp.ftp.upload.ooyala.com`

Note:

- For faster uploads, Ooyala recommends that you use the servers in the region to which you are closest.
- There are times when our servers are under heavy load and may not respond as quickly. Should you want more responsive uploads, you can use the alternative FTP servers.

Manifest File Formats

When ingesting videos, you can use manifest files to specify extra metadata (including the video title, description, flight times, labels, and custom metadata), as well as thumbnail and closed caption files. Manifest files are in either XML or CSV format.

Note: This topic applies to both OIS v2.5 and OIS v1. Certain sections, where indicated, apply to just one version. OIS v1 has been disabled. Customers using OIS v1 should switch to OIS v2.5.

OIS v1 only: You must let Ooyala know whether you will use XML or CSV format in your manifest files. Each requires its own set of credentials. Contact your Ooyala representative for details.

Note: Backlot does not begin processing any of the videos until the XML or CSV manifest file is received. As a result, it is important to choose a batch upload strategy. If your videos must be available quickly (e.g., news distribution), you should create very small batches or even batch them individually (one CSV or XML file per video).

In addition to the static CSV and XML manifest file formats, you can use filters to apply dynamic manifest criteria to assets. See *Dynamic Manifests* on page 26.

CSV Manifest File

When ingesting videos, you can use CSV manifest files to specify extra metadata (including the video title, description, flight times, labels, and custom metadata), as well as thumbnail and closed caption files. You can use CSV manifest files when you ingest content with FTP or Aspera.

Note: This topic applies to both OIS v2.5 and OIS v1. Certain sections, where indicated, apply to just one version. OIS v1 has been disabled. Customers using OIS v1 should switch to OIS v2.5.

The CSV format of the metadata file supports fewer features than does the XML format. The CSV format allows you to specify thumbnails (preview images) and custom metadata to associate with a video. The XML format supports these features as well as the association of the names of existing defined of ad sets.

CSV Manifest Column Headings

The following fields are the first line (column headings) of your CSV file, with actual data fields following on subsequent rows (one row per video).

Note: You can specify these columns in any order, and you can omit columns that you do not need, as long as the data in the data rows match the headings exactly.

Column Heading	Description	Required/ Optional
`video`	Filename of the video or the URL where the video is located. The URL can include `http://` or `https://`.	required
`title`	Name or title of the video.	optional
`thumbnail`	Filename of the thumbnail or the URL where the thumbnail is located.	optional

Column Heading	Description	Required/ Optional
content_type	(OS v2.5 only) Content type (video or remote asset). If the filename specified in the video column is a URL where the video is located, and if you want to define a remote asset, change the value of the content_type column (from video, the default) to content_type=remoteasset. See *Ingesting Remote Assets* on page 32.	required (remote assets only)
description	Description of the video.	optional
hosted_at	Permanent URL where you embed the video. Maps to the value of the hosted_at property for a remote asset. **Note:** After a remote asset has been created, its propagation to the various CDNs might be delayed 60 seconds or more. If you request a remote asset too soon after its creation, the results will be cached by the CDNs, which might take several minutes to clear. Best practice: after creation, wait 30 or 60 seconds, query with the Backlot API [GET] /v2/assets/*asset_id* route, and after retrieving the remote asset's embed code (content ID or asset ID), then proceed to embed the asset.	optional
flight_start_time	The start time when the asset can be played, in UTC. Example: 2011-06-01T00:00:00Z	required (only if flight_end_timeis specified)
flight_end_time	The end time when the asset can be played, in UTC. Example: 2011-07-01T00:00:00Z	optional
durationInMs	(OS v2.5 only) The duration representing the length of the video (in milliseconds). Required when content_type=remoteasset.	required (for remote assets only)
labels	One or more labels, separated by commas. For example: /sports,/sports/jogging. For background, see *Labels*.	optional
metadata	Custom metadata for the video. You can have a column for each type of metadata (for example, metadata:*type_1* metadata:*type_2* and so on). When using this element with VR 360 video, set the column name to vr360type and the metadata value to stereoscopic(side-by-side) or monoscopic. See *Custom Metadata* for format requirements.	optional
embed_code	(Reserved). Content ID in Backlot of the video asset.	reserved
id	Maps to the created asset's external ID property: a custom identifier you define that you can use instead of the content ID. See *Updating Asset Metadata Using CSV or XML Manifest Files* on page 25.	optional
subTitle	Specifies a closed caption file (supported for DFXP files only). One of the following: • Filename of the closed caption file you uploaded. Example: <media:subTitle href="caption-1.dfxp"/> • URL where the closed caption file is located. The URL can include http:// or https://. Example: <media:subTitle href="http://ooyala.com/captions/caption-1.dfxp"/> See *Ingesting Closed Caption Files* on page 29.	optional

Column Heading	Description	Required/ Optional
`subTitle:lang`	(optional) - Language of the closed caption file. Two-letter language code. For valid values, see *Supported Closed Captions*.	optional
`subTitle:frameRate`	(optional) - Frame rate of the closed caption file. Units are in Frames Per Second (FPS). Default is 30. Older closed caption files might be 24.	optional
`profileguid`	ID of the processing profile, which is a group of encodes that define the formats a master video must be converted into during transcoding. Contact your Ooyala support representative for details.	optional

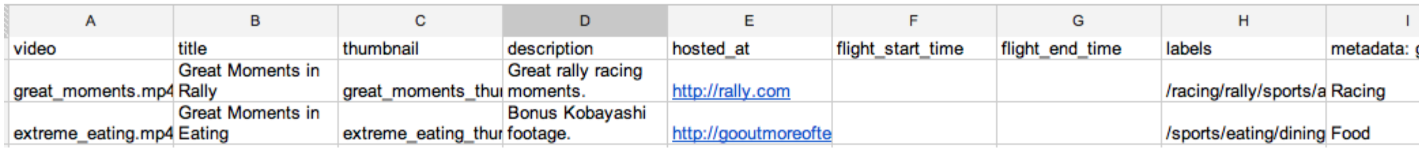
Example CSV Manifest File (OIS v2.5)

This example is for a VR 360 video file.

```
video,title,description,labels,hosted_at,id,thumbnail,subTitle,flight_start_time,flight_
  test.mp4,Test CSV,This is an example,"test,example,csv",http://
www.ooyala.com,my_test_guid_csv,my_thumbnail.jpg,my_caption.dfxp,2016-01-01T15:15:00-07
by-side)
```

Example CSV Manifest Data in Excel (OIS v1) (Deprecated)

Note: OIS v1 has been disabled. Any customers using OIS v1 must switch to OIS v2.5.



A	B	C	D	E	F	G	H	I
video	title	thumbnail	description	hosted_at	flight_start_time	flight_end_time	labels	metadata:
great_moments.mp4	Great Moments in Rally	great_moments_thu	Great rally racing moments.	http://rally.com			/racing/rally/sports/a	Racing
extreme_eating.mp4	Great Moments in Eating	extreme_eating_thu	Bonus Kobayashi footage.	http://gooutmoreofte			/sports/eating/dining	Food

Creating a CSV Manifest File

Keep the following in mind when generating a CSV metadata file:

- If a text value in a data field has a comma (,), you must enclose the field in double quotes (").
- If a text value in a data field has a double quote (") in it, you must enclose the field in double quotes and escape the double quote character in the text field with a second quote (e.g., "She said, ""Have a nice day.""")

If you are creating your CSV metadata file from Excel, it automatically handles escaping.

To create a CSV metadata file:

1. Upload one or more videos.
2. Open a spreadsheet program, such as Microsoft Excel.
3. Add the column headings to the first row.
4. Add a row for each video.
5. Save the file as a CSV. For example, you might select **Save as** from the **File** menu. When prompted, select the comma-separated value file type.
6. With your upload method of choice, upload the CSV file as you uploaded the video(s). See either *FTP* or *Aspera*.

XML Manifest File

When ingesting videos, you can use XML manifest files to specify extra metadata (including the video title, description, flight times, labels, and custom metadata), as well as thumbnail and closed caption files. You can use XML manifest files when you ingest content with FTP, Aspera, or MRSS.

Note: This topic applies to both OIS v2.5 and OIS v1. Certain sections, where indicated, apply to just one version. OIS v1 has been disabled. Customers using OIS v1 should switch to OIS v2.5.

XML Manifest Elements

You can use the following elements in the XML manifest file.

XML Tag	Description	Required/Optional
`<channel>`	Contains one or more `<item>` elements.	required
`<item>`	Represents a single video to ingest. You can specify multiple videos (`<item>` tags) in an XML manifest file. Contains the remaining tags in this table.	required
`<media:content>`	Specifies a video to ingest. One of the following: • Filename of a video that you uploaded. • For ingestion via MRSS (see *Ingesting from a Remotely Hosted MRSS Feed* on page 15), the URL where the video is located. The URL can include `http://` or `https://`. For a remote asset, use `<ooyala:remoteasset>` instead.	required (except for remote assets)
`<media:title>`	Name or title of the video.	optional
`<media:thumbnail>`	Specifies a thumbnail (preview image). One of the following: • Filename of a thumbnail you uploaded. • URL where the thumbnail is located. The URL can include `http://` or `https://`.	optional
`<media:subTitle>`	Specifies a closed caption file (supported for DFXP files only). See *Ingesting Closed Caption Files* on page 29. Attributes: • `href` (required) - One of the following: • Filename of the closed caption file you uploaded. Example: `caption-1.dfxp` • URL where the closed caption file is located. The URL can include `http://` or `https://`. Example: `http://ooyala.com/captions/caption-1.dfxp` `lang` (optional) - language of the closed caption file. Two-letter	optional

XML Tag	Description	Required/Optional
	code. See *Supported Closed Captions*. • `frameRate` (optional) - Frame rate of the closed caption file. Units are in Frames Per Second (FPS). Default is 30. Older closed caption files might be 24. **Note:** If specified, it is the customer’s responsibility to provide valid values for `lang` and `frameRate`. Valid values are added to the DFXP file. Non-compliant values are ignored. Either way, the closed caption file is added into Backlot and linked to the asset.	
`<media:description>`	Description of the video.	optional
`<link>`	Permanent URL where you embed the video. Maps to the created asset's `hosted_at` property.	optional

`<dcterms:valid>`	Flight times representing the start and end times when the asset can be played. Example:
	
	`<dcterms:valid>start=2011-06-28T15:15:00-07:00;`
	
	` end=2022-12-01T16:00:00-07:00;scheme=W3C-DTF`
	`</dcterms:valid>`
		optional

`<ooyala:labels>`	One or more labels, separated by commas. Example: `/sports,/sports/jogging`. For background, see *Labels*.	optional

`<media:keywords>`	Another way to define labels (without a forward slash). The contents, separated by commas, will be added as labels to the account (for example, `sports,sports/jogging`). A new label will be created only if a label with the same name does not already exist.
	
	For example, for the following `<media:keywords>` tag, the labels `label1` and `label2` would be created. A label named `parentlabel` would also be created, with the label `childlabel` nested inside it.
	
	`<media:keywords>label1,`
	` label2,`
		optional

XML Tag	Description	Required/Optional
	`parentlabel/ childlabel</ media:keywords>` **Note:** The contents of `<media:keywords>` are ignored when the `<ooyala:labels>` tag is included in the XML manifest file or MRSS.	
`<ooyala:embedcode>`	(Reserved) Content ID of the video asset as specified in Backlot.	reserved
`<ooyala:profileguid>`	ID of the processing profile, which is a group of encodes that define the formats a master video must be converted into during transcoding. Contact your Ooyala support representative for details.	optional
`<ooyala:metadata>`	Custom metadata for the video. You can have an element for each type of metadata. Example: `<ooyala:metadata name="internal_id">`, `<ooyala:metadata name="category">`, and so on. When using this element with VR 360 video, set the `name` attribute to `"vr360type"` and the content of the element to `stereoscopic(side-by-side)` or `monoscopic`. Example: `<ooyala:metadata name="vr360type">monoscopic</ooyala:metadata>`. See *Custom Metadata* for format requirements.	optional
`<ooyala:remoteasset>`	Specifies a remote asset to ingest. A remote asset is a video that is hosted somewhere other than Backlot. Specify the following attributes: • `durationInMs`: media duration (in milliseconds) • `url`: URL of the remote asset. The URL can include `http://` or `https://`. This setting maps to the value of the `hosted_at` property for a remote asset. Example: `<ooyala:remoteasset durationInMs='5000' url="http://your_website/your_remote_assets.mp4" />`	required (remote assets only)

XML Tag	Description	Required/Optional
	Note: After a remote asset has been created, its propagation to the various CDNs might be delayed 60 seconds or more. If you request a remote asset too soon after its creation, the results will be cached by the CDNs, which might take several minutes to clear. Best practice: after creation, wait 30 or 60 seconds, query with the Backlot API `[GET] /v2/assets/`*`asset_id`* route, and after retrieving the remote asset's embed code (content ID or asset ID), then proceed to embed the asset.	
`<guid>`	External ID (identifier) for the video, such as the embed code. See *Updating Asset Metadata Using CSV or XML Manifest Files* on page 25. **Example:** `<guid isPermaLink="false">12345678910</guid>`	optional

Example XML Manifest File (OIS v2.5)

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0" xmlns:media="http://search.yahoo.com/mrss/"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:fh="http://purl.org/syndication/history/1.0"
  xmlns:ooyala="http://www.ooyala.com/mrss/">
  <channel>
    <item>
      <media:content url="test.mp4" ></media:content>
      <media:subTitle href="my_caption.dfxp"></media:subTitle>
      <media:thumbnail url="my_thumbnail.jpg" filesize="35712"></
media:thumbnail>
      <media:title>Test XML</media:title>
      <media:description>This is an example</media:description>
      <ooyala:labels>/test,/example,/xml</ooyala:labels>
      <dcterms:valid>start=2016-01-01T15:15:00-07:00;
        end=2022-01-01T16:00:00-07:00</dcterms:valid>
      <guid isPermaLink="false">my_test_guid_xml</guid>
      <ooyala:profileguid>57d5a5a10f654fe79ef954dc8d29a108</
ooyala:profileguid>
    </item>
  </channel>
</rss>
```

Example XML Manifest File (OIS v1) (Deprecated)

Note: OIS v1 has been disabled. Any customers using OIS v1 must switch to OIS v2.5.

In the following example, the XML specifies metadata for two videos. For the first video, the file, closed caption and thumbnail are located on an upload server and are pulled from that server. For the second video, the file, closed caption and thumbnail were already uploaded to an Ooyala server. Also, the second video has flight times.

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0" xmlns:media="http://search.yahoo.com/mrss/"
     xmlns:dcterms="http://purl.org/dc/terms/"
     xmlns:fh="http://purl.org/syndication/history/1.0"
     xmlns:ooyala="http://www.ooyala.com/mrss/">

     <channel>
       <item>
          <media:content url="http://mysite.com/upload/lacrosse_70.mov" />
          <media:thumbnail url="http://mysite.com/upload/
lacrosse_70_previewimage.jpg" />
          <media:subTitle href="http://mysite.com/upload/caption-1.dfxp" />
          <media:title>Sports that Really Exist: Lacrosse</media:title>
          <media:description>My description of my video</media:description>
          <ooyala:labels>/sports/lacrosse,/hobbies/lacrosse</ooyala:labels>
          <link>http://mysite.com/videos/sports/real_sports.html</link>
          <ooyala:metadata name="video_ID">70</ooyala:metadata>
          <ooyala:metadata name="season_number">1</ooyala:metadata>
       </item>

       <item>
          <media:content url="curling_71.mov" />
          <media:thumbnail url="curling_71_previewimage.jpg" />
          <media:subTitle href="caption-1.dfxp" />
          <media:title>Sports that Really Exist: Curling</media:title>
          <media:description>My description of my video</media:description>
          <ooyala:labels>/sports/curling,/hobbies/curling</ooyala:labels>
          <link>http://mysite.com/videos/sports/real_sports.html</link>
          <ooyala:metadata name="video_ID">71</ooyala:metadata>
          <ooyala:metadata name="season_number">1</ooyala:metadata>

          <dcterms:valid>start=2011-06-28T15:15:00-07:00;
               end=2022-12-01T16:00:00-07:00;scheme=W3C-DTF
          </dcterms:valid>
       </item>
     </channel>
</rss>
```

Creating an XML Manifest File

To create an XML metadata file:

1. Upload your videos and other content (thumbnails, closed caption files, and so on).
2. Open a text editor, preferably one that validates XML.
3. Add the required XML headers. For example:

   ```
   <?xml version="1.0" encoding="utf-8"?>
   <rss version="2.0" xmlns:media="http://search.yahoo.com/mrss/"
        xmlns:dcterms="http://purl.org/dc/terms/"
        xmlns:fh="http://purl.org/syndication/history/1.0"
        xmlns:ooyala="http://www.ooyala.com/mrss/">
   ```

4. Create a `channel` container.
5. Create an `item` entry for each video, containing the desired elements from the table above.

In the following example, the XML specifies metadata for two videos. For the first video, the file and thumbnails are located on an upload server and are pulled from that server. For the second video, the file and thumbnail were already uploaded to an Ooyala server. The second video also has flight times.

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0" xmlns:media="http://search.yahoo.com/mrss/"
     xmlns:dcterms="http://purl.org/dc/terms/"
     xmlns:fh="http://purl.org/syndication/history/1.0"
     xmlns:ooyala="http://www.ooyala.com/mrss/">

     <channel>
           <item>
            <media:content url="http://mysite.com/upload/
lacrosse_70.mov" />
            <media:thumbnail url="http://mysite.com/upload/
lacrosse_70_previewimage.jpg" />
            <media:title>Sports that Really Exist: Lacrosse</media:title>
            <media:description>My description of my video</
media:description>
            <ooyala:labels>/sports/lacrosse,/hobbies/lacrosse</
ooyala:labels>
            <link>http://mysite.com/videos/sports/real_sports.html</link>
            <ooyala:metadata name="video_ID">70</ooyala:metadata>
            <ooyala:metadata name="season_number">1</ooyala:metadata>
           </item>

           <item>
            <media:content url="curling_71.mov" />
            <media:thumbnail url="curling_71_previewimage.jpg" />
            <media:title>Sports that Really Exist: Curling</media:title>
            <media:description>My description of my video</
media:description>
            <ooyala:labels>/sports/curling,/hobbies/curling</
ooyala:labels>
            <link>http://mysite.com/videos/sports/real_sports.html</link>
            <ooyala:metadata name="video_ID">71</ooyala:metadata>
            <ooyala:metadata name="season_number">1</ooyala:metadata>
            <!-- set flight times -->
            <dcterms:valid>start=2011-06-28T15:15:00-07:00;
                end=2022-12-01T16:00:00-07:00;scheme=W3C-DTF
                </dcterms:valid>

           </item>
     </channel>
</rss>
```

6. Save the file as XML. Be sure to specify a unique name. If you specify two files with the same name and upload one while the other is processing, the second one might be ignored.
7. With your upload method of choice, upload the XML manifest file as you uploaded the video(s). See either *FTP* or *Aspera*.

Updating Asset Metadata Using CSV or XML Manifest Files

Use the `id` column (in CSV manifest files) or the `guid` element (in XML manifest files) to specify your own identifiers for your ingested content. You can later use those identifiers to update the asset, including its metadata.

The `id` column (in a *CSV Manifest File* on page 17) and the `guid` element (in an *XML Manifest File* on page 19) work as follows:

- If you do not explicitly set the field when the file is first ingested, the identifier is the embed code (content ID or asset ID) assigned by the system.

- If you do set the field in the initial upload, you need to provide it in the CSV or XML manifest files for later updates to a previously uploaded file.
- If the CSV or XML manifest file includes the field with the external ID or embed code (content ID or asset ID) of a previously uploaded video file, the system updates the asset's metadata.
- If the CSV or XML manifest file references a video file, the system looks for that video file locally in the customer's directory. If the system cannot find it, the entry is skipped.

Note: If you are not including your external identifiers, do *not* include these fields with a null value. Null values cannot be searched for later. Omit the field from your manifest file.

Dynamic Manifests

If you are using Microsoft Azure Media Services to store uploaded assets, you can use a predefined filter to generate a customized, dynamic manifest for transcoding. Filters are server-side rules that are most commonly used to specify a subset of audio and video renditions to provide the best viewing experience on a given device. For example, you can set up a filter to request that when an asset is transcoded, only HD and SD renditions are created.

A dynamic manifest is created when a viewer requests to stream a video that has a filter associated with it. This is in comparison to other types of Ooyala manifest files, which are uploaded along with the assets and remain the same for all viewing requests (see *CSV Manifest File* on page 17 or *XML Manifest File* on page 19). An asset can have both a static manifest file and a dynamic manifest filter.

Dynamic manifests are useful when you need more flexibility than what can be defined in the asset’s default manifest file. With a dynamic manifest, you can:

- Better control the customer viewing experience on various device types
- Reduce CDN usage

SETTING UP DYNAMIC MANIFESTS

To enable dynamic manifests, contact your Ooyala support representative and provide the desired filter criteria. The Ooyala team configures the filter for your account and sends you the name of the filter.

You can specify the desired output in terms of:

- Bitrate (single bitrate or range of bitrates)
- Bitrate for audio stream only
- Bitrate for video stream only

USING FILTERS

To put a filter into effect, include the filter name when you embed the Ooyala Player on a web page or when you make an iOS SDK or Android SDK call to display video in a mobile app.

Example: Web Player

```
var playerParam = {"dynamicFilters": "filterA,filterB,filterC"};
OO.Player.Create(DIV, embed_code, playerParam);
```

Example: Android SDK

```
List<String> dynamicFilters = Arrays.asList("filterA", "filterB",
 "filterC");

Options options = new
 Options.Builder().setDynamicFilters(dynamicFilters).build();
player = new OoyalaPlayer(pcode, new PlayerDomain(domain), null, options);
```

Example: iOS SDK with Swift

```
let options: OOOptions = OOOptions()!
options.dynamicFilters = ["filterA", " filterB", "filterC ", " filterD "]
player = OOOoyalaPlayer(pcode: pcode, domain: domain, embedTokenGenerator:
 nil, options: options)!
```

Example: iOS SDK with Objective-C

```
OOOptions *options = [OOOptions new];
options.dynamicFilters = [[NSArray alloc]
initWithObjects:@"filterA",@"filterB",@"filterC",nil];
OOOoyalaPlayer *ooyalaPlayer = [[OOOoyalaPlayer alloc]
initWithPcode:self.pcode domain:[[OOPlayerDomain alloc]
initWithString:self.playerDomain] options:options];
```

LIMITATIONS

You can include multiple filters on a single asset, up to a maximum of three (3) filters per asset.

ERROR HANDLING

If the filter name in the embed parameter or SDK call does not match any defined filter (for example, in case of a typographical error in your code), the filter is not applied and the video transcodes normally as if no filter had been requested.

Ingestion Log

INGESTION LOG REST API (OIS V2.5)

If you ingest using FTP, Aspera, or MRSS, you can use the Ingestion Log REST API (API) to search for log information based on various criteria. You can filter by time period (basic and manifest-based users) or status (manifest-based user only). For details, see the *Ooyala Ingestion Log API documentation*.

INGESTION LOG AND ERROR MESSAGES (OIS V1) (DEPRECATED)

Note: OIS v1 has been disabled. Any customers using OIS v1 must switch to OIS v2.5.

When you use FTP or Aspera to ingest content, Ooyala creates a log file in the home directory called `ooyala.log`.

Note: The `ooyala.log` is used with OIS v1 only.

You can view this file to check the status of uploads.

Example Success Message

The following snippet shows what is recorded in the ingestion log for a successful upload.

```
     .
     .
     .
   2012-10-31 02:25:48 INFO        importing "http://
d18tka3ecu2l5z.cloudfront.net/39206_MOV01E_1228612704000.flv"
   2012-10-31 02:26:04 INFO        fetching http://
d18tka3ecu2l5z.cloudfront.net/39206_MOV01E_1228612704000.flv
   2012-10-31 02:26:07 INFO        succeeded: received 41351347.0 bytes
   2012-10-31 02:26:14 INFO        assigned embed_code
 o2b2lnNjq28XhRGgF26cVEJP1iF2hv2R
```

```
    2012-10-31 02:26:21 INFO        updating metadata for
  o2b2lnNjq28XhRGgF26cVEJP1iF2hv2R
    2012-10-31 02:26:22 INFO        success
    .
    .
    .
```

Error Messages

The following table describes possible error messages.

Error Message	Description
expecting X bytes; got Y bytes	The downloaded file wasn't the correct size; resend the metadata file.
remote server said 404 or remote server said 505	OIS couldn't find the file on your server; check that the video file is there and send the metadata file again.
remote server said ###	An error occurred on your server; check your server and resend the metadata file.
hash mismatch: expecting ..., got ...	The file was corrupt; resend the metadata file.
failed to upload thumbnail	The movie was ingested, but the thumbnail upload failed. You can manually fix this by uploading a new thumbnail in the Backlot UI or upload the video, thumbnail, and metadata file for that video again.
error fetching feed ...	There was a problem retrieving the MRSS feed; make sure the MRSS server is working properly.
failed: workflow error	There was a problem with the content you provided. You can manually fix this by checking the state and configuration of the video in the Backlot UI or upload the video, thumbnail, and metadata file for that video again.
failed: system busy; try again	There was an issue with the uploading the video. You can manually fix this by checking the state and configuration of the video in the Backlot UI or upload the movie, thumbnail, and metadata file for that video again.
failed: internal error	Retry the upload. If it is not fixed within five minutes, file a ticket on the *Customer Portal*.

INGESTING CLOSED CAPTION FILES

You can ingest closed caption files via Backlot or the Ooyala Ingestion Service (OIS).

Note: This topic applies to both OIS v2.5 and OIS v1. Certain sections, where indicated, apply to just one version. OIS v1 has been disabled. Customers using OIS v1 should switch to OIS v2.5.

WAYS TO INGEST CLOSED CAPTION FILES

- *Uploading a Closed Caption File in Backlot*
- *Ingesting Closed Captions Using XML Manifest Files*
- *Ingesting Closed Captions Using CSV Manifest Files*
- *Working with Closed Captions*

SUPPORTED FORMATS FOR CLOSED CAPTION FILES (OIS INGESTION)

You can upload closed caption files that use any of the following file formats.

Note: Playback on Ooyala mobile apps (apps developed using the Ooyala iOS and Android SDKs) requires the TTML (formerly DFXP) format. Player V4 Web supports additional closed-caption formats across all browser environments, including browsers on iOS and Android devices.

Extension	Description
`.TTML` (formerly DXFP)	TTML (Timed Text Markup Language file. Generally, a closed caption file will have a .dfxp (Distribution Format Exchange Profile) extension. For details, see *Supported Closed Caption DFXP (now TTML) Format*.
`.SCC`	SCC (Scenarist Closed Caption) file.
`.SRT`	WebSRT (Web Subtitle Resource Tracks) file.
`.VTT`	WebVTT (Web Video Text Tracks) file.

XML MANIFEST FORMATS FOR CLOSED CAPTION FILES

Use the following elements and attributes in the *XML Manifest File* on page 19.

XML Tag	Description
`<media:subtitle>`	Specifies a closed caption file (supported for DFXP files only). See *Ingesting Closed Caption Files* on page 29. Attributes: • `href` (required) - One of the following: • Filename of the closed caption file you uploaded. Example: `caption-1.dfxp` • URL where the closed caption file is located. The URL can include `http://` or `https://`. Example: `http://ooyala.com/captions/caption-1.dfxp` `lang` (optional) - language of the closed caption file. Two-letter code. See *Supported Closed Captions*. • `frameRate` (optional) - Frame rate of the closed caption file. Units are in Frames Per Second (FPS). Default is 30. Older closed caption files might be 24. **Note:** If specified, it is the customer's responsibility to provide valid values for `lang` and `frameRate`. Valid values are added to the DFXP file. Non-compliant values are ignored. Either way, the closed caption file is added into Backlot and linked to the asset.

XML Tag	Description
`<media:hash>`	(OIS v1 only) (Optional) Within the `<media:subTitle>` element, you can optionally include the `hash` element to specify a cryptographic hash value and the hash function algorithm that was used to generate the hash value. Use the syntax shown in the following example. `<media:hash algo=”md5”>5d42be7f301dad5acf42d565fd651857</media:hash>` where • `algo` is `md5` (hash function) • `hash` value is `5d42be7f301dad5acf42d565fd651857`

XML Manifest Example for Closed Captions (OIS v2.5)

```
<rss>
 <channel>
  <item>
   <media:subTitle href="filename or URL">closedCaption.dfxp lang="jp"
 frameRate="24"</media:subTitle>
  </item>
 </channel>
</rss>
```

XML Manifest Example for Closed Captions (OIS v1) - Simple Example (Deprecated)

Note: OIS v1 has been disabled. Any customers using OIS v1 must switch to OIS v2.5.

```
<rss>
  <channel>
    <item>
      <media:subTitle href="filename or URL">
        <media:hash algo="chosen hash algorithm">hash value</media:hash>
      </media:subTitle>
    </item>
  </channel>
</rss>
```

XML Manifest Example for Closed Captions (OIS v1) - Longer Example (Deprecated)

Note: OIS v1 has been disabled. Any customers using OIS v1 must switch to OIS v2.5.

```
<rss version="2.0" xmlns:media="http://search.yahoo.com/mrss/"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:fh="http://purl.org/syndication/history/1.0"
  xmlns:ooyala="http://www.ooyala.com/mrss/">
  <channel>
   <item>
    <media:content url="movie1.mp4">
     <media:hash algo="md5">23ac5a7c88378522f95483e5e9ac44b1</media:hash>
    </media:content>
    <media:title>Movie #1</media:title>
    <media:description>Movie #1 Description</media:description>
    <media:subTitle href="closedCaption.dfxp">
     <media:hash algo="md5">5d42be7f301dad5acf42d565fd651857</media:hash>
    </media:subTitle>
    <link>http://www.ooyala.com/</link>
    <guid isPermaLink="false">c92643a33463910098fa0b9eae7b973c-3</guid>
   </item>
  </channel>
</rss>
```

CSV MANIFEST FORMATS FOR CLOSED CAPTION FILES

Use the following column headings for ingesting closed caption files via a *CSV Manifest File* on page 17:

Column Heading	Description
`subTitle`	Specifies a closed caption file (supported for DFXP files only). One of the following: • Filename of the closed caption file you uploaded. Example: `<media:subTitle href="caption-1.dfxp"/>` • URL where the closed caption file is located. The URL can include `http://` or `https://`. Example: `<media:subTitle href="http://ooyala.com/captions/caption-1.dfxp"/>` See *Ingesting Closed Caption Files* on page 29.
`subTitle:lang`	(optional) - Language of the closed caption file. Two-letter language code. For valid values, see *Supported Closed Captions*.
`subTitle:frameRate`	(optional) - Frame rate of the closed caption file. Units are in Frames Per Second (FPS). Default is 30. Older closed caption files might be 24.

In the CSV file:

- The first line of the CSV file must contain a `subTitle` column heading. Optionally, specify `subTitle:lang` and / or `subTitle:frameRate`.
- Subsequent lines must contain the corresponding data: filename or URL where the closed caption file is located (as applicable), language code (optional), and frameRate (optional).

Note: If specified, it is the customer's responsibility to provide valid values for `subTitle:lang` and `subTitle:frameRate`. Valid values are added to the DFXP file. Non-compliant values are ignored. Either way, the closed caption file is added into Backlot and linked to the asset.

The following example CSV column headings line includes all three column headings:

```
video,title,description,hosted_at,labels,thumbnail,subTitle,subTitle:lang,subTitle:frame
my_video.mp4,My Video Title,This is my video,http://
www.ooyala.com,"label1,label2",my_preview.jpg,my_caption.dfxp,en,24,,my_unique_id
```

INGESTING REMOTE ASSETS

A remote asset is a piece of content that is hosted outside of Backlot. Instead of uploading, transcoding, and storing the content in Backlot, you simply upload its URL and other descriptive information so that Backlot can find the content when needed. For an introduction, see *Remote Assets*. You can ingest remote assets to Backlot using the following ways:

- Backlot UI (see *Remote Assets*)
- Backlot API (see *Remote Assets*)
- Media RSS (MRSS) (see *Ingesting Remote Assets via MRSS* on page 32)

For an overview, see *Remote Assets*.

INGESTING REMOTE ASSETS VIA MRSS

You can add or update remote assets using Media RSS (MRSS).

For more information, see *Remote Assets*.

When using MRSS to add or update a remote asset, you identify the asset as remote by specifying the `<ooyala:remoteasset>` tag (instead of `<media:content>`) in the MRSS feed item.

Note: To decide whether to add or update a remote asset, Ooyala uses the specified `<guid>` (or, if no `<guid>` is specified, the specified `<url>`) to determine whether the remote asset currently exists (update) or not (add).

Adding a Remote Asset

The following example code creates a remote asset (the specified <guid> does not exist).

```
<rss xmlns:media="http://search.yahoo.com/mrss/" xmlns:dcterms="http://
purl.org/dc/terms/" xmlns:fh="http://purl.org/syndication/history/1.0"
 xmlns:ooyala="http://www.ooyala.com/mrss/" version="2.0">
  <channel>
    <item>
   <ooyala:remoteasset durationInMs='1000' url="http://mysite.com/
RCTTestAssets/mrss/assets/Test.mp4"/>
   <media:thumbnail url="http://mysite.com/test.jpg"/>
   <media:subTitle href="http://mysite.com/test.dfxp"/>
   <media:title>Test</media:title>
   <media:description>Test</media:description>
   <ooyala:labels>/Test</ooyala:labels>
   <link>http://mysite.com/videos/sports/test_0010.html</link>
   <!-- set flight times -->
   <dcterms:valid>
    start=2011-06-28T15:15:00-07:00;
 end=2022-12-01T16:00:00-07:00;scheme=W3C-DTF
   </dcterms:valid>
   <guid>Test001</guid>
    </item>
  </channel>
</rss>
```

Updating a Remote Asset

For an existing remote asset, you can update the following fields via MRSS:

- `media:title`

- `media:description`
- `ooyala:subTitle`
- `ooyala:metadata`
- `dcterms:valid`
- `media:thumbnail`
- `ooyala:labels`

When you submit the MRSS feed item, if it contains updates to any of these fields in the `<item>` tag, the system updates the remote asset with the applicable changes.

ADDING A REMOTE ASSET IN BACKLOT

Before you can manage a Remote Asset, you must add it to Backlot.

To add a remote asset:

1. Log in to the *Backlot UI*.
2. Click **ADD NEW CONTENT** and select **Remote Asset**.
   The **Add Remote Asset** dialog box appears.
3. Enter the URL to the remote asset.
4. If the remote asset is a VR 360 Video (see *VR 360 Videos*), select the **VR 360** checkbox and select the type of content (**Monoscopic** or **Stereoscopic (side-by-side)**.
5. Click **OK**.
   Backlot adds the remote asset.
6. Change the default name in the **TITLE** field and provide a description in the **DESCRIPTION** field.

ADDING A REMOTE ASSET USING THE BACKLOT REST API

Before you can manage a Remote Asset, you must add it to Backlot.

Note: For more information about Backlot REST API commands, see the *Backlot API Reference*.

To add a remote asset:

Use the `/v2/assets` route.

The following example creates the "My Remote Asset" remote asset.

```
[POST]/v2/assets{
   "name":"My Remote Asset",
   "asset_type":"remote_asset",
   "duration":120000,
   "stream_urls":{
      "flash":"http://mydomain.com/my_flash_file.flv",
      "iphone":"http://mydomain.com/iphone_compatible_file.mp4"
   }
}
```

Backlot returns a response similar to the following.

```
{
   "asset_type":"remote_asset",
   "duration":120000,
   "name":"My Remote Asset",
   "preview_image_url":null,
   "created_at":"2011-05-16T20:35:53+00:00",
   "embed_code":"Y1NWZnMjq-DPsM2",
   "stream_urls":{
      "flash":"http://mydomain.com/my_flash_file.flv",
      "iphone":"http://mydomain.com/iphone_compatible_file.mp4"
   },
   "time_restrictions":null,
   "updated_at":"2011-06-02T00:08:58+00:00",
   "external_id":"myExternalId",
   "hosted_at":null,
   "original_file_name":null,
   "description":null,
   "status":"live"
}
```

Note: Try out the code samples using your account credentials in the Ooyala Scratchpad. To launch the Scratchpad, go to Ooyala *API Scratchpad*. For information about using the Scratchpad, see *Practice Making Requests with the Scratchpad*.

The remote asset is successfully added.

Note: You can add VR 360 videos as remote assets. To do so, you must include an additional query parameter (`vr360type` attribute) that specifies the type of VR 360 video (monscopic or stereoscopic). See *Uploading VR 360 Content* for details.

CONTENT MIGRATION

This documentation will walk you through a standard content migration, your options, and best practices.

If you have an extensive video library, and require assistance in moving your content to Ooyala, you will benefit most from Ooyala's Content Migration Services. When self-migration (i.e. via Backlot Web upload, FTP, Aspera, APIs) is not an option, a Professional Services-assisted migration makes sense.

Content migration is a three step process:

1. Prepare your source and manifest files.
2. Upload your source and manifest file.
3. Once your videos are Live, publish them to your website, or start swapping your old embeds with new ones.

STEP 1: PREPARE YOUR SOURCE AND MANIFEST FILES

You should always start with the best quality source material available. Ooyala has specific recommendations at *Quality of and Recommendations on Source Material (Video and Audio)*. Once you have your source files ready, you should start working on your manifest file.

The term *metadata* is used here in two ways:

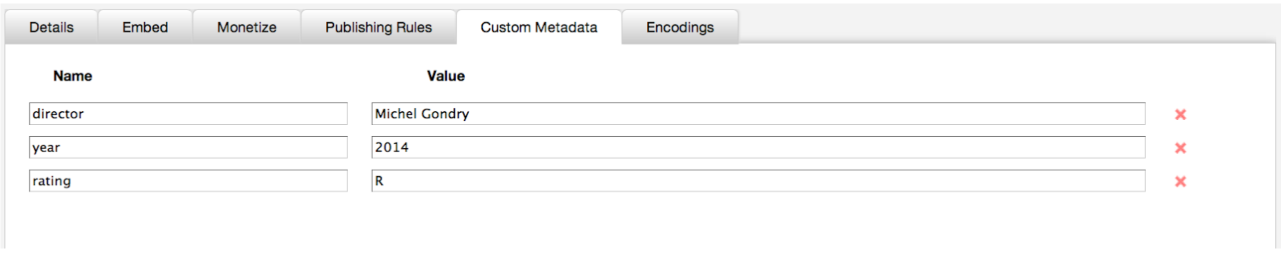
- "Custom Metadata" means up to 100 name-value pairs used to store data describing the video itself. For example, in a movie library, you could find the following name:value pairs.

```
"director" : "Michel Gondry",
    "year" : "2004",
  "rating" : "R"
```

- "Labels" are tags that you can use to group videos for their display, analytics, syndication etc. Any given video can be tagged with one or more labels, and every label can hold many videos. For example, in a movie library you could label your videos according to their production studio.

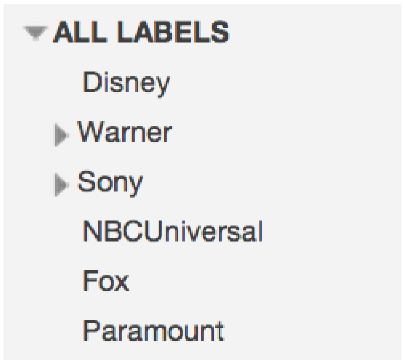
If you are planning on making bulk assignments of either custom players, ad sets, or syndication rules, consider that the easiest way of multi-selecting assets is to group them by Labels. For example, it might make sense to add and assign to assets labels like: "Blue Player", "Green Player", "Only Colombia", "Kid Friendly" etc ...

Custom Metadata. Continuing with our movie library example, once your manifest file is processed, you can log into Backlot and find assigned Custom Metadata on each asset's Custom Metadata Tab.

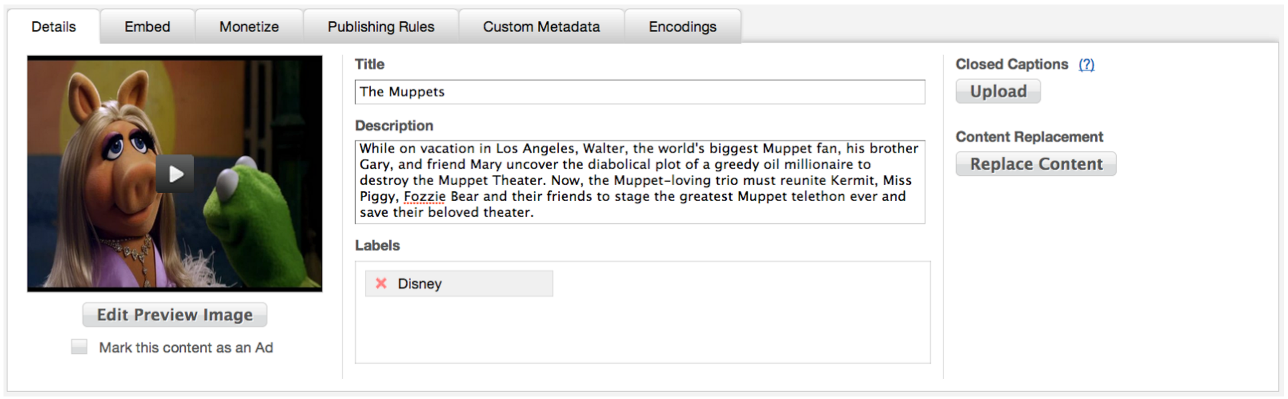


Name	Value
director	Michel Gondry
year	2014
rating	R

Label Hierarchy. See your Label hierarchy in the Manage window.



Label Assignments per Asset. See Label assignments per asset in its Details Tab.



Title

The Muppets

Description

While on vacation in Los Angeles, Walter, the world's biggest Muppet fan, his brother Gary, and friend Mary uncover the diabolical plot of a greedy oil millionaire to destroy the Muppet Theater. Now, the Muppet-loving trio must reunite Kermit, Miss Piggy, Fozzie Bear and their friends to stage the greatest Muppet telethon ever and save their beloved theater.

Labels

Disney

Use create manifest files in either XML or CSV. See *Manifest File Formats* on page 17 for details.

STEP 2: UPLOAD YOUR SOURCE AND MANIFEST FILE

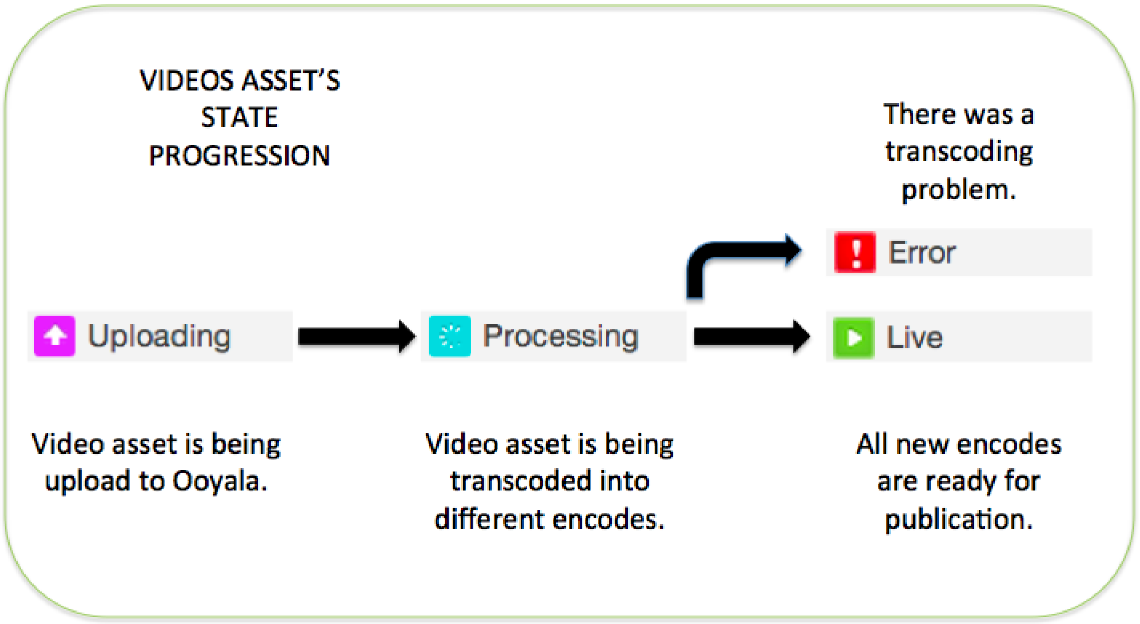
Ooyala supports several *Ways to Ingest Content* on page 10:

- Backlot UI
- Ooyala Import Services (OIS), with or without metadata about the videos in either comma-separated value (CSV) or XML format, via the following transport mechanisms:
  - FTP
  - Aspera
  - MRSS
- Backlot API, specifically the /v2/assets routes

See *Ingestion* for details.

STEP 3: ONCE YOUR VIDEOS ARE LIVE, PUBLISH THEM TO YOUR WEBSITE

Throughout the content migration process, video assets pass through several states:


VIDEOS ASSET'S STATE PROGRESSION
There was a transcoding problem.
Error
Uploading
Processing
Live
Video asset is being upload to Ooyala.
Video asset is being transcoded into different encodes.
All new encodes are ready for publication.

If you encounter a video in an Error state, contact Technical Support for troubleshooting. Videos in the Live state are ready for publishing.

To learn how to embed a player on a web page and its structure, see *Player*.

If you are planning on swapping old players, you will need to:

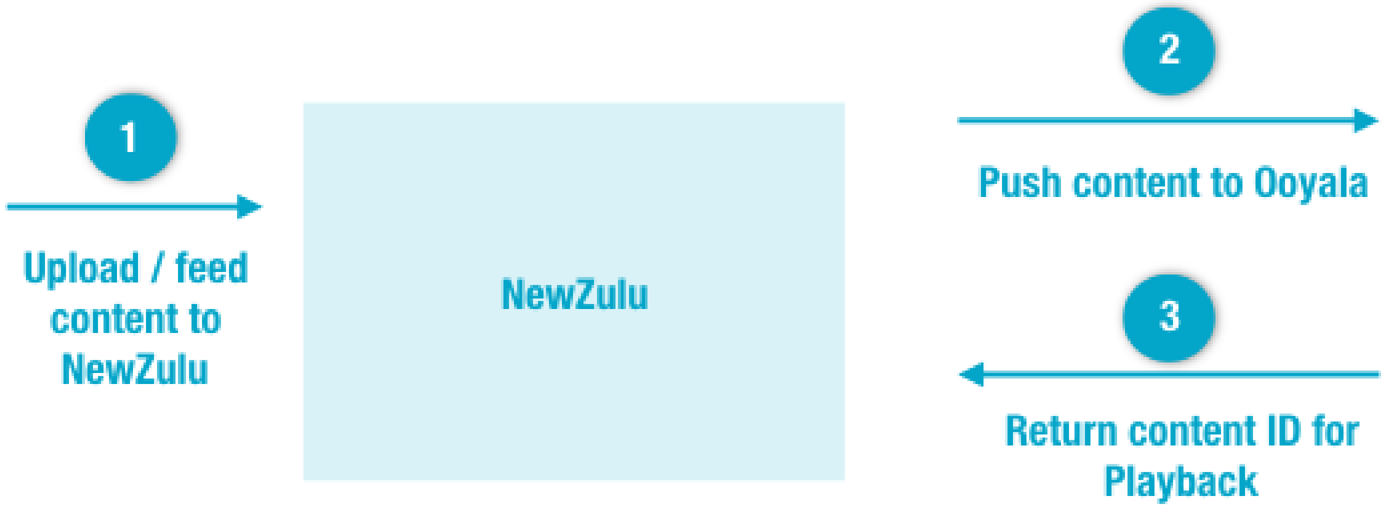
1. Find and identify all your old player embeds.
2. Assign each video a unique identifier that you can tie back with its old embeds, and save that identifier on either the `id` field of the CSV manifest file or in the `guid` field of the XML manifest file.
3. Once the Content is Live, you will need to construct a file mapping your new embed identifiers with the old one using our APIs. Also, if your site uses dynamic pages, playlists, or channels, you will need to re-program them using Ooyala's APIs. See *Backlot REST API*.

INTEGRATE WITH NEWZULU

The Ooyala integration with *NewZulu's* FileMobile and Media Factory products allow users to seamlessly combine NewZulu's content collection, curation, and workflow capabilities with Ooyala analytics, advertising, and video services.

ABOUT OOYALA-NEWZULU INTEGRATION

NewZulu's FileMobile product provides desktop and mobile users with an easy to use, responsive video upload and integrated playback experience. Once integration is set up:


1
Upload / feed content to NewZulu
NewZulu
2
Push content to Ooyala
3
Return content ID for Playback

- **NewZulu ingest**. Users upload videos and other files via widgets (such as their *Uploader widget*) and *APIs* to NewZulu's Media Factory. Media Factory also collects streamed content (live streams, social media, RSS, and so on).
- **Ooyala Ingest**. Media Factory provides configurable workflow capabilities to push collected content (along with any associated metadata) to the Ooyala platform via *Ingesting with FTP / FTPS* on page 11. Users can configure trigger options to push content to Ooyala automatically, to push only reviewed and approved content, or to send moderated content individually. In addition to standard metadata (for example, information about the user who uploaded the content), Media Factory can pass on custom metadata.
- **Content ID for Playback**. Once ingested, Ooyala returns the asset's external content ID to Media Factory. This reference allows for interactive playback of the content using NewZulu widgets (such as their *Gallery Widget*) or their API.

INTEGRATION SETUP INSTRUCTIONS

To set up integration, you'll need:

- an Ooyala account
- a NewZulu account

In Media Factory, you will need to configure how to push content to Ooyala. This requires:

- your Ooyala account information (login credentials, API key, and API secret)
- FTP configuration settings

Refer to the *NewZulu's integration instructions* for setup details. For more information about NewZulu products, refer to their *Developer Documentation Portal*.

INTEGRATION TECHNICAL SUPPORT

For technical support of Ooyala-NewZulu integrations:

Contact	For
NewZulu Technical Support	Issues associated with NewZulu widgets, APIs, and Media Factory, including content uploads and collection, curation, approvals, workflow, and push to the Ooyala platform.
Ooyala Customer Portal.	Post-publishing issues, including video playback and quality, Ooyala analytics, and ad serving.

INTEGRATE WITH AVID TECHNOLOGIES

This topic provides an overview of the Ooyala integration with *Avid Technology*.

ABOUT OOYALA-AVID INTEGRATION

For Ooyala customers, integration with Avid Technologies allows Avid users to push content directly from their *Media Suite* into Backlot.

INTEGRATION INSTRUCTIONS

To integrate, you must have:

- Avid account
- Ooyala account

Setup instructions are easy - you simply make Ooyala a publishing target for content:

1. Launch Avid's *Media | Distribute tool*
2. Add a profile of type Ooyala, add your API Key and API Secret, and select a transcoding profile.

For details, see Avid's Ooyala integration instructions ("Configuring an Ooyala Account," pages 102-105) in Avid's *Media | Distribute Installation and Configuration Guide*.

INTEGRATION TECHNICAL SUPPORT

For technical support of Ooyala-Avid integrations:

Contact	For
Avid Technical Support	Issues associated with Avid Media Suite and the push workflow to the Ooyala platform.
Ooyala Customer Portal	Post-publishing issues, including video playback and quality, Ooyala analytics, and ad serving.

INGESTION RELEASE NOTES

MRSS FEED VALIDATOR FOR OIS V2.5 (2018-01-04)

The Ooyala MRSS Feed Validator has been updated with a new DTD and now works with OIS v2.5. See *Ingesting from a Remotely Hosted MRSS Feed* on page 15.

SUPPORT FOR VR 360 UPLOADS VIA OIS (2017-12-20)

This release introduces support for uploading VR 360 videos into Backlot using OIS. Uploading a VR 360 video is similar to ingesting a flat video, with the addition of metadata that is unique to VR 360 - label the file as VR 360 and specify the VR 360 type (monoscopic or stereoscopic).

- You can set VR 360 file metadata in the manifest file (see *Manifest File Formats* on page 17).
- If you are not using a manifest file, indicate the file type by setting file metadata with the `ffmpeg` command-line utility (see *Ingesting with FTP / FTPS* on page 11 or *Ingesting with Aspera* on page 13).

For an overview, see *VR 360 Videos*.

DASHBOARD ENHANCEMENTS AND IN-REGION INGESTION (2017-10-12)

- **Performance enhancements** increase the average speed of content being ingested. Content is more quickly available for playback.
- **In-region ingestion and transcoding** using Azure Media services, reducing overall ingestion and transcoding time and keeping content in the same region as the customer. See *About Ingesting Content* on page 4.
- **New search feature in Ingestion Dashboard.** Look up an asset that is being ingested and view its status.
- **More information visible to more user roles in Ingestion Dashboard.** It is no longer required to log in with Upload Only privileges to see the Ingestion Dashboard (Ingestion Feed Activity tab). All user roles can now see this dashboard. The amount of information that can be seen is determined by the credentials used to log in to Backlot, as shown in the following table:

User Role	Added Functionality
Upload Only	View the status of ingest jobs you initiated.
All other user roles	View the Ingestion Dashboard. Track the ingestion status of all content uploaded into the Backlot account from OIS using all credentials.

See *Viewing the Ingest Log in Backlot*.

NEW LANGUAGE AND FRAMERATE OPTIONS FOR CLOSED CAPTION INGESTION (2017-06-08)

If you use manifest files to ingest closed caption files via the Ooyala Ingestion Service (OIS), you can now optionally specify the language and framerate of the closed caption file. See *Ingesting Closed Caption Files* on page 29 for details.

NEW INGESTION DASHBOARD (2017-05-10)

The Backlot UI added an **Ingestion Feed Activity** tab that displays a read-only log of files submitted for ingestion to Backlot via the Ooyala Ingestion Service (OIS). This dashboard displays the name and type of the submitted file, its processing status, and other related information. You can filter the view to narrow search results. To see this tab, you must log into Backlot as a user with **upload only** access. For details, see *Viewing the Ingest Log in Backlot*.

XML MANIFEST VALIDATION IN OIS V2.5 (2017-03-06)

For OIS 2.5 users, the system has been enhanced to allow unsupported elements and tags in XML manifest files. As long as the required Ooyala elements and tags (ooyala.dtd) are present, the system logs allows the ingestion process to continue without the process failing. Previously, unsupported tags in a manifest would cause an ingestion failure. See *XML Manifest File* on page 19.

SEE ALSO

- *Ingestion*
- *Introducing the Online Video Platform*