Media Delivery Core (MDC)

Use cases for Asset Ordering, Delivery and Tracking

CONTENTS

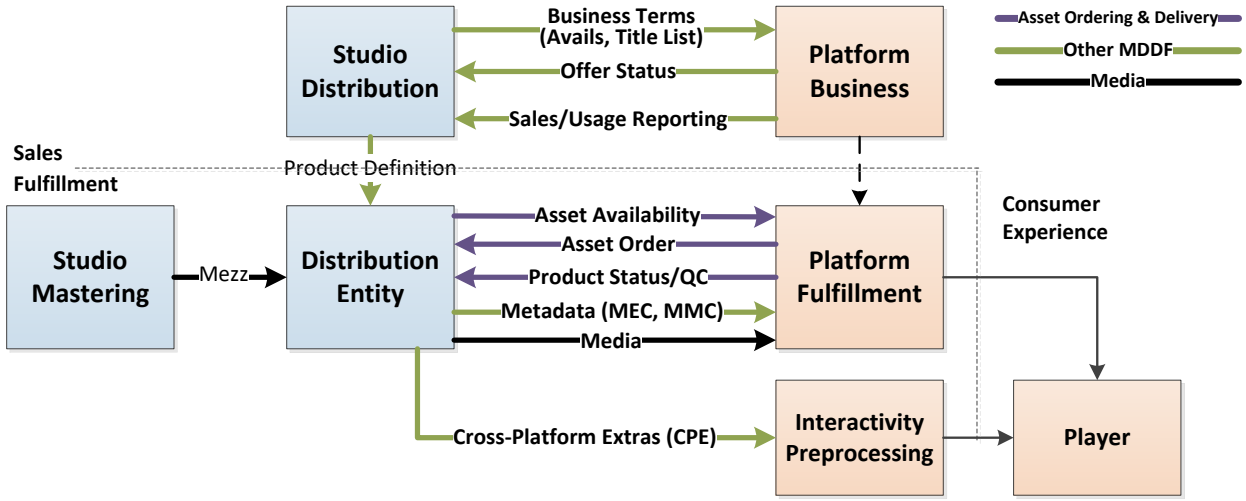
REVISION HISTORY

Version	Date	Description
1.0	December 14, 2019	First Release

1 INTRODUCTION

This specification is designed to simplify implementations of Asset Ordering and Delivery. The Asset Ordering and Delivery specification (www.movielabs.com/md/delivery) supports a broad range of use cases, without specific instructions for any given use case. This document provides specific recommendations and instructions for a variety of high-priority use cases.

This document is part of a series of specifications collectively referred to as the MovieLabs Digital Distribution Framework (MDDF). This includes document Media Manifest Core (MMC), Media Manifest, Media Entertainment Core (MEC), Common Metadata, Avails, Common Ratings, QC Vocabulary others. For more information on MDDF, see www.movielabs.com/md. The full specification supports more complex delivery use cases, interactivity and other applications.



This document was produced a collaboration between MovieLabs, The Entertainment Merchant's Association (EMA), the Digital Entertainment Group (DEG) and the members of these organizations; collectively referred to as the Digital Supply Chain Alliance (DSCA).

1.1 Document Organization

This document is organized as follows:

1. Introduction—Background, scope and conventions
2. General Encoding Guidelines
3. Asset Planning
4. Product Status

1.2 Document Notation and Conventions

The document uses the conventions of Common Metadata [CM].

1.3 Normative References

[Delivery]	TR-META-AOD, Asset Ordering, Delivery, and Tracking, v1.0, www.movielabs.com/md/delivery
[Manifest]	TR-META-MMM MovieLabs Media Manifest Metadata, v1.9, http://www.movielabs.com/md/manifest
[CM]	TR-META-CM MovieLabs Common Metadata, v2.8, http://www.movielabs.com/md/md
[MMC]	TR-META-MMC, *Media Manifest Delivery Core*, v.1.9, www.movielabs.com/md/mmc
[MEC]	*Media Entertainment Core (MEC) Metadata, version 2.8.* www.movielabs.com/md/mec
[AVAILS]	*Entertainment Merchant's Association (EMA) Avails.* www.movielabs.com/md/avails
[CDR]	*Content Delivery Requirements (CDR), www.movielabs.com/md/cdr*
[EIDR-TO]	*EIDR Technical Overview*, November 2010. http://eidr.org/technology/#docs

All Common Metadata and Media Manifest references are included by reference.

1.4 Informative References

[Practices]	MDDF Best Practices: www.movielabs.com/md/practices
[CPE]	Cross-Platform Extras (CPE): www.movielabs.com/cpe

1.5 XML Namespaces

This document uses the following namespaces:

- md – Defined in Common Metadata [CM]
- manifest – Defined in Media Manifest [Manifest]
- delivery – Defined in Asset Ordering and Delivery [Delivery]

1.6 Identifiers

Identifiers must be universally unique. Recommended identifier schemes may be found in Common Metadata [CM] and in DECE Content Metadata [DECEMD].

The use of Entertainment Identifier Registry identifiers (www.eidr.org) is strongly encouraged. Please see [EIDR-TO].

Best practices for identifiers can be found on www.movielabs.com/md/practices.

1.7 Status

This specification is completed and is ready for implementation. Although tested, we anticipate that additional implementation experience will yield recommendation for changes. Implementers should anticipate one or more revisions.

Reasonable measures will be taken to ensure changes are backwards compatible. See Backwards Compatibility Best Practices in [CM]

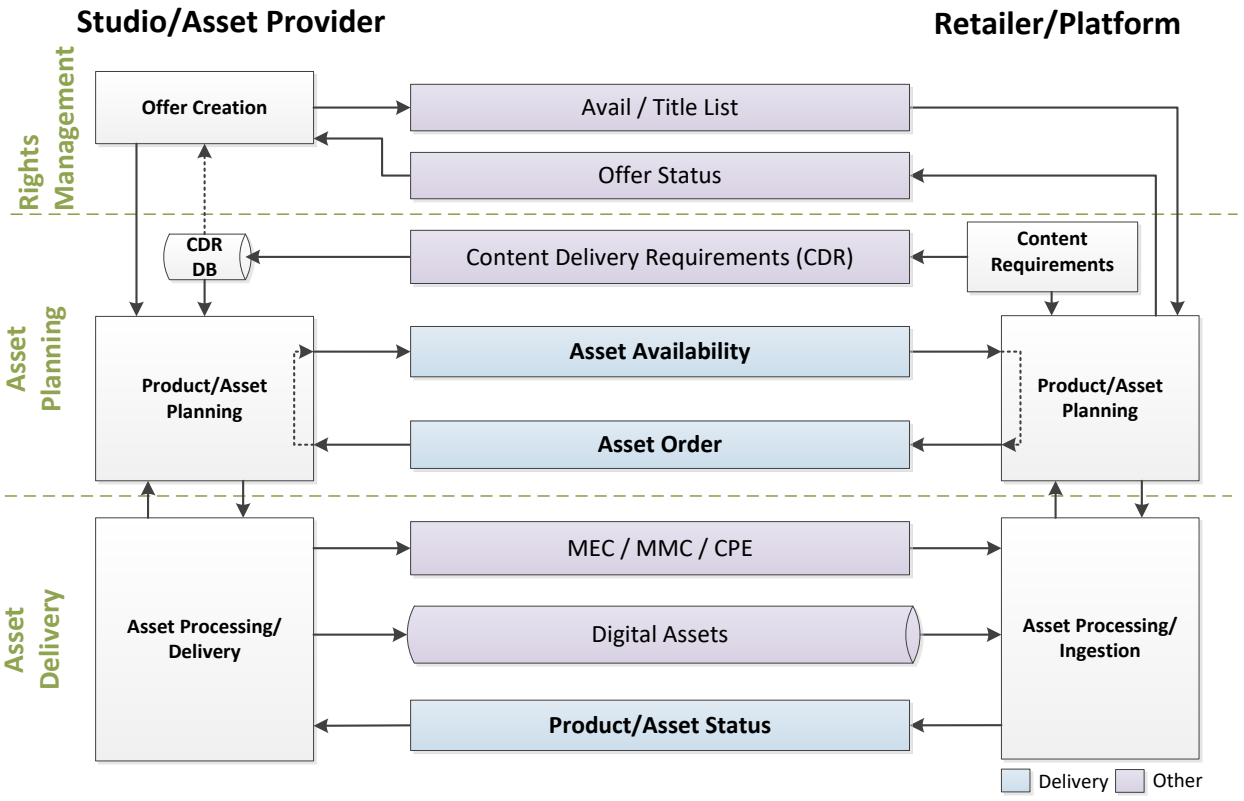
1.8 Using this Document

This document is intended to be the starting point for understanding the Asset Ordering Delivery and Status process, much like Media Manifest Core (MMC) is a starting point for Media Manifest and Media Entertainment Core (MEC) is a starting point for Common Metadata.

MDC is intended to be used with MEC, MMC and CPE which provide specific practices for their respective specifications.

1.8.1 Workflow Model

The following model represents a multitude of workflows, rather than any specific workflow. This document is focused on Asset Availability, Asset Order and Product Status.



It is unlikely that anyone would use everything in this uber-workflow. In practice, a workflow would use only selected data objects, and within those data objects, only a small number of specific features.

1.8.2 General Encoding Guidelines

The General Encoding Guidelines section provides instruction on how to use key building blocks that are used across the different messages. This includes source and destination information, how to reference an offer (i.e., what the content is fulfilling), and reference content.

These guidelines are referenced from specific use cases. When we get into use case details, we focus on particulars to those use cases. Refer to the general guidelines for encoding the rest of the message.

1.8.3 Use Cases

This document is focused on two categories of use cases:

- Asset Planning – How content providers and platforms come to agreement on what content will be delivered
- Product Status – How platforms provide content providers with information about the status of asset delivery. This ranges from general status updates, to detailed error reporting.

2 GENERAL ENCODING GUIDELINES

2.1 Encoding Source and Destination

Message have top-level Source and Destination elements to describe where messages are coming from and where they are going.

As described in [Delivery], DeliveryPlatform-type and DeliveryPublisher-type are used to describe these message sources and destinations. Source and destination may include service providers.

2.1.1 Direct communications studio to platforms

When messages are sent between studios and platforms directly (i.e., no service providers), it is often unnecessary to included Source and Destination. However, it is always best practice to include them.

Here is a source that's a platform.

```
<delivery:Source>
      <delivery:Platform>
         <md:DisplayName>craigsmovies.com</md:DisplayName>
      </delivery:Platform>
      <delivery:ServiceProvider>
         <md:DisplayName>Fix My Movies SP</md:DisplayName>
      </delivery:ServiceProvider>
</delivery:Source>
```

2.1.2 Service Providers

When service providers are generating or receiving a message, ServiceProvider should be included.

When ServiceProviders are intermediaries, Platform and Publisher should also be included to ensure source or destination is understood. Consider a service provider acting on behalf of multiple studios. That service provider must include both Publisher or Platform, and ServiceProvider in any message sent.

In this example, craigsmovies.com is using the "Fix My Movies SP" service provider. Fix My Movies SP generated the following message on behalf of craigsmovies.com with a destination of Sofa Spud Films.

```
<delivery:Source>
      <delivery:Platform>
         <md:DisplayName>craigsmovies.com</md:DisplayName>
      </delivery:Platform>
      <delivery:ServiceProvider>
         <md:DisplayName>Fix My Movies SP</md:DisplayName>
      </delivery:ServiceProvider>
</delivery:Source
<delivery:Destination>
      <delivery:Publisher>
         <md:DisplayName>Sofa Spud Films</md:DisplayName>
      </delivery:Publisher>
</delivery:Destination>
```

2.1.3 Contact Information

Contact Information is provided in the Contact. It is recommended that Contact be included to facilitate direct communication in case there are problems. This is the primary contact point when Instruction/ExceptionFlag is asserted.

```
<delivery:Contact>
   <md:Name>Jane C. Contact</md:Name>
   <md:PrimaryEmail>jane@sofaspudsfilms.com</md:PrimaryEmail>
   <md:Phone>555-555-5555</md:Phone>
</delivery:Contact>
```

2.2 Scope

Scope defines the context of the object. Less formally, the object is being generated to convey information or a request about a set of assets *related to something*. That something is the scope.

2.2.1 Scope for Avails and Title Lists

When the scope of an object is relative to an Avails or Title List, ALID or AlternateID is used. Once can be more specific by using TransactionID. If using Excel Avails and Title List, TransactionID gets the AvailID data.

This example uses ALID. That means the scope of this update is all assets associated with that ALID (i.e., all Avails or Title List entries that include that ALID).

```
<delivery:Scope>
      <delivery:ALID>md:alid:eidr-s:AD07-310C-C59D-6785-C63A-G</delivery:ALID>
</delivery:Scope>
```

2.2.2 Scope for Titles

Let's say you want to define assets in terms of a particular movie or TV episode. The best method is to use the EIDR ID. Alternatively, AlternateID can be used.

EIDRURN is used for deliveries associated with a title (generally an edit).

```
<delivery:Scope>
      <delivery:EIDRURN scope="Edit">urn:eidr:10.5240:AD07-310C-C59D-6785-C
      </delivery:EIDRURN>
</delivery:Scope>
```

2.2.3 Constraining Scope

If the request is more specific, elements are provided to restrict scope.

It's generally a good idea to keep scope as simple as possible. Put the language and format profile descriptions deeper in the asset definitions. If scope is narrower than the asset definitions, the object will be invalid (although not necessarily detected by the validator).

Region or ExcludedRegion can be used to narrow the scope when referring to assets specific to a territory. In the following example, the scope is everything associated with that ALID in the US. If a particular language is not provided in the US, it would not be within scope.

```
<delivery:Scope>
      <delivery:ALID>md:alid:eidr-s:AD07-310C-C59D-6785-C63A-G</delivery:ALID>
      <delivery:Region>
         <md:country>US</md:country>
      </delivery:Region>
</delivery:Scope>
```

Scope can be narrowed using Language (i.e., what language does apply to) and FormatProfile (e.g., SD, HD, UHD).

```
<delivery:Scope>
      <delivery:ALID>md:alid:eidr-s:AD07-310C-C59D-6785-C63A-G</delivery:ALID>
      <delivery:Region>
         <md:country>US</md:country>
      </delivery:Region>
      <delivery:Language>en-us</delivery:Language>
      <delivery:FormatProfile>HD</delivery:FormatProfile>
</delivery:Scope>
```

Note that any combination of Region (ExcludedRegion), Language and FormatProfile can be used—whatever makes sense.

2.3 Referring to content, tracks, files, etc.

Throughout these specs, it is necessary to refer to content. Sometimes it's very general (e.g., "All content associated with offer *xyz*"). Sometimes it's very specific (e.g., "00:31:25.1-00:35:00.00 in audio track *abc*"). And, sometimes it's somewhere in between (e.g., "Best French dub for title *pqr*"). We supply models for the range from general to specific.

Which one should be used depends on how it's being used. For example, when ordering content, it's usually a more general request. When referring to a media error, it's usually specific.

This section provides some example that are referred to in specific use cases. Note that in the spirit of "core" spec, these do not cover all use cases. A much wider range of use cases can be supported by the spec.

In the following sections we talk about

- Referencing Tracks, files, etc. – This is something that already exists and can be things like track references and track identifiers.
- Content – In this context, we refer to content regardless of its encoding or instantiation. Or more to the point, content that cannot be referenced by file names, track references, track identifiers, etc.

2.3.1 Referencing a specific track by reference (identifier)

The primary object for referencing media is delivery:DeliveryObjectReference-type. You'll find this in objects such as AssetOrder/Asset/Reference and ProductStatus/AssetStatus/AssetReference.

Note that when referencing tracks, it is often to include other identifiers, especially EIDR, to provide more context.

2.3.1.1 Referencing objects using MDDF identifiers

By MDDF identifiers, we are referring to any object defined as part of the MovieLabs Digital Distribution Framework. This includes Common Metadata/MEC, MediaManifest/MMC, Avails and Title List, and so forth.

Media Manifest has various mechanisms for referencing content. If referencing content in Media Manifest, the most robust mechanism is to use the Manifest Track Identifiers. These are found in ObjectReference/TrackID.

In TrackID you can include a reference to relevant track. If it's a video track, you include VideoTrackID.

Let's take a simple example. Let's say there is a problem with a specific audio track.

```
<delivery:ProductStatus>
      ...
   <delivery:AssetStatus>
      <delivery:AssetReference>
         <delivery:MDDFID>
            <delivery:AudioTrackID>md:audiotrackid:eidr-x:AD07-310C-C59D-6785-C63A-
G:audio.fr</delivery:AudioTrackID>
         </delivery:MDDFID>
...
```

2.3.1.1.1 Simple Reference

This can be used to reference any MDDF object. For example, if the problem is in metadata, it looks almost identical.

```
<delivery:ProductStatus>
      ...
   <delivery:AssetStatus>
      <delivery:AssetReference>
         <delivery:MDDFID>
            <delivery:ContentID>md:cid:eidr-s:AD07-310C-C59D-6785-C63A-G</delivery:ContentID>
         </delivery:MDDFID>
...
```

2.3.1.1.2 More than one reference

It might be necessary to reference more than one track. Let's say, for example, that there is an A/C sync issue (i.e., audio and video are out of sync within a presentation). In this case, one would want to reference at least the Presentation and the audio track. Note that this assumes the video is the reference.

```
<delivery:ProductStatus>
      ...
   <delivery:AssetStatus>
      <delivery:AssetReference>
         <delivery:MDDFID>
            <delivery:PresentationID>md:cid:eidr-x:AD07-310C-C59D-6785-C63A-
G:main</delivery:PresentationID>
         </delivery:MDDFID>
         <delivery:MDDFID>
            <delivery:AudioTrackID>md:audiotrackid:eidr-x:AD07-310C-C59D-6785-C63A-
G:audio.fr</delivery:AudioTrackID>
         </delivery:MDDFID>
...
```

2.3.1.2 Referencing non-MDDF identifiers

Here is an example referencing an Apple Vendor Identifier using DeliveryObjectReference-type

```
<delivery:ProductStatus>
      ...
   <delivery:AssetStatus>
      <delivery:AssetReference>
         <delivery:TrackIdentifier>
            <md:Namespace>/package/video/vendor_id</md:Namespace>
            <md:Identifier>00000000012345</md:Identifier>
         </delivery:TrackIdentifier>
...
```

2.3.1.3 Referencing objects in IMF

There are some special features for IMF. These are defined in Common Metadata [cm], Section 5.18, and extended in the Asset Ordering and Delivery Spec.

When referring to a track, it is necessary to reference both the CPL and the Virtual Track ID. Purists will not that since the Virtual Track Identifier is a UUID, is unique, it is an unambiguous reference to the track. However, we're trying to make things easy to find and the CPL ID is one additional hint.

A reference might look something like this:

```
<delivery:IMFRef>
   <manifest:CPLID>urn:uuid:de6d2644-e84c-432d-98d5-98d89271d082</manifest:CPLID>
   <delivery:VirtualTrackID>urn:uuid:f3e86156-007d-4649-a985-
e468df5a0f37</delivery:VirtualTrackID>
</delivery:IMFRef>
```

Generally, an OPL reference isn't necessary unless there is reference to a particular encoding of the track. However, in these cases there is usually a better reference for the encoded track.

2.3.2 Referencing tracks by description

In practice we often refer to tracks descriptively; for example, "The French dub for *xxx*". An example where this is found is in Asset Ordering: AssetOrder/Asset/Description.

There are two parts to this. From, the example above, the first is *xxx*, which is the work for which the French dub is needed. The second part is the characteristics of the asset itself: French dub. Characteristics can range from the general, as just stated, or a bit more specifically ("The French dub 5.1") or even more specific ("The French Dub, 5.1, DD+").

So, there are always two parts: work identification and characteristics.

2.3.2.1 Identification

Identification is an identifier (no surprise there) that unambiguously references a work. For various reasons, this is preferably done with an EIDR. The EIDR unambiguously defines either and abstraction or edit corresponding to the work in question. One can also use other forms of identification.

Note that when we get into specifics like encoding an Asset Order, there are other identifiers that have relevance, such as ALID. Note also that we do not support manual title matching, so you're not going to see title strings here.

2.3.2.1.1 Identifying using EIDR

EIDR objects can be referenced using EIDRURN. Note that this has an @structuralType attribute that allows you to indicate whether it's an Abstraction, Edit or Manifestation.

It is frequently useful to include EIDR, even when using other forms of identification.

The following example is a partial Asset Order object

```
<delivery:Reference>
   ...
   <delivery:EIDRURN scope="Edit">urn:eidr:10.5240:AD07-310C-C59D-6785-C
   </delivery:EIDRURN>
</delivery:Reference>
```

Note that the EIDR is always expressed in URN format: https://tools.ietf.org/html/rfc7972. The definition of @scope is in Common Metadata [CM], Section 2.1.2.

2.3.2.2 Characteristics

When we talk about characteristics, we're talking metadata. So, the rich metadata definitions from Media Manifest are borrowed for this purpose. These can be found in md:DigitalAssetSet-type.

The challenge is defining exactly the metadata you need. The simple answer is to keep it as simple as possible.

Let's go back to the French Dub example. The following example is from AssetOrder/Asset/Description. Note that we're explicitly using French from France (fr-FR), not Quebecois (fr-CA).

```
<delivery:Description>
   <md:Audio>
      <md:Type>Primary</md:Type>
      <md:Language>fr-FR</md:Language>
   </md:Audio>
</delivery:Description>
```

Let's say we want to be more specific and specify 5.1:

```
<delivery:Description>
   <md:Audio>
      <md:Type>Primary</md:Type>
         <md:Language>fr-FR</md:Language>
         <md:Channels>5.1</md:Channels>
   </md:Audio>
</delivery:Description>
```

Let's finish the example with French dub in 5.1 DD+. Dolby Digital Plus is more formally Enhanced AC-3, encoded, per definition in Common Metadata, as "E-AC-3".

```
<delivery:Description>
   <md:Audio>
      <md:Type>Primary</md:Type>
```

```
      <md:Encoding>
         <md:Codec>E-AC-3</md:Codec>
      </md:Encoding>
      <md:Language>fr-FR</md:Language>
      <md:Channels>5.1</md:Channels>
   </md:Audio>
</delivery:Description>
```

This can be extended to include any descriptions in Common Metadata Digital Asset Metadata; but only if you need it. Only specify what you care about!

Note that certain asset requirements, particularly language requirements, are implicit in Avails and Title Lists (www.movielabs.com/md/avails). Default delivery requirements are generally defined in written platform specifications. An alternative method for specifying detailed platform requirements can be found a draft specification called *Content Delivery Requirements (CDR).*

2.3.2.3 Language and Component

This is a special narrow case of referencing by Characteristics that is used exclusively in ProductStatus/ObjectStatus. This is described under Product Status in Section 4.

2.4 Terms

There are various Terms objects throughout the spec (e.g., Asset/TechnicalTerms, Asset/BusinessTerms, AssetOrder/TermsAcrossAssets). This is a flexible structure used to add terms that are bilateral, or terms that are a little to obscure to include in the schema.

```
<delivery:TermsAcrossAssets>
   <delivery:TechnicalTerms termName="AssetQuality">BetterThanAverage</delivery:TechnicalTerms>
</delivery:TermsAcrossAssets>
```

Another example of a Term would be whether delivery is flat files or component.

```
<delivery:AssetDisposition>
   <delivery:TechnicalTerms termName="DeliveryModel">FlatFile</delivery:TechnicalTerms>
</delivery:AssetDisposition>
```

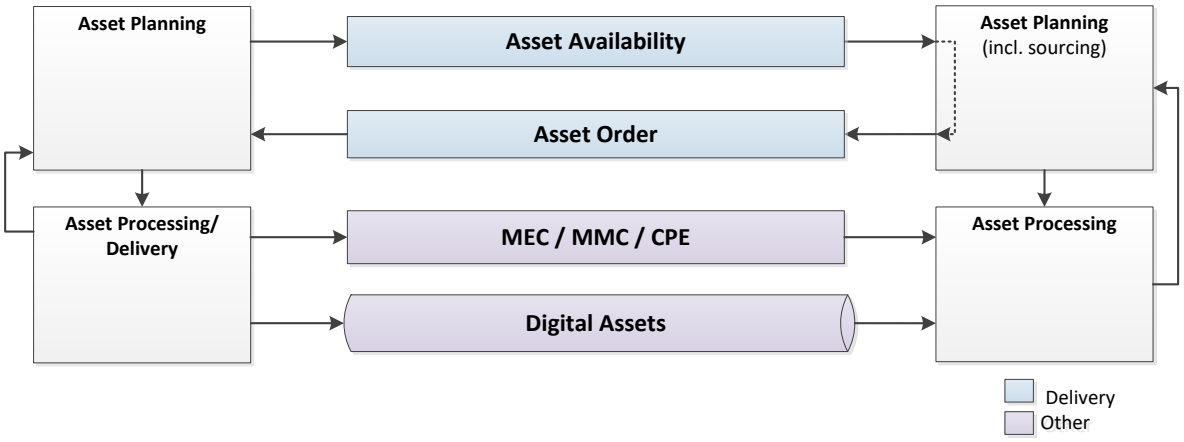
3 ASSET PLANNING

Asset Planning is where content providers (or service providers) and platforms (or service providers) agree on the specific set of assets to be delivered.

This process assumes there are agreements outside of Asset Ordering and Delivery. Ideally, information is delivered using standard methods (see Content Delivery Requirements [CDR][1]). Often, these requirements are specified in contracts, or some form of specification. Many large platforms have help sites with their requirements.

Using these requirements, if any, as a point of departure, Asset Availability and Asset Order are used to reach agreement on what assets will be delivered.

Note that MEC/MMC/CPE and Digital Assets are shown to illustrate the exchange of media files and the data that accompanies them.



3.1 General Information

3.1.1 Push, Pull and Negotiated models

We describe use cases in terms of ‘push’, ‘pull’ and ‘negotiated’. In the Push model, the content provider informs the platform what it will be receiving. This generally applies to retailers who say, “Send us everything” or platforms that have pre-negotiated titles (e.g., SVOD or AVOD service with a Title List [Avails] defining assets). The push begins with an AssetAvailability message.

Note that if platforms just want whatever studios send, they can skip this process entirely and go straight to sending a Media Manifest (see [MMC]).

The pull model applies when the platform is requesting assets. This might be in response to an Avail or Title List that references specific assets. The pull begins with an AssetOrder object. In some cases, the Platform might already have suitable assets. In this case, those assets could be excluded from the AssetOrder.

[1] CDR is part of the broader model described in Section 1.8.1.

The negotiated model starts with an AssetAvailabilty or AssetOrder, but has AssetOrder or AssetAvailability responses, respectively. This can be used to refine the asset list. For example, let's say a platform needs assets that are not in AssetAvailability. They can respond with an AssetOrder that includes the missing assets. The content provider can then respond with an AssetOrder that has additional information, such as dates and cost. The platform can then confirm or reject the new information. It's likely some of the actual negotiation happens through human contact, but these messages provide the automation and record keeping that ensure intent is fulfilled.

3.1.2 Scope

In AssetAvailability and AssetOrder, Scope indicates the general context of the status. Guidance for encoding Scope can be found in Section 2.2.

If it refers to an Avails or Title List, use ALID or AlternateID. To be more specific, use TransactionID. If using Excel Avails and Title List, TransactionID gets the AvailID data.

Region or ExcludedRegion can be used to narrow the scope when referring to assets specific to a territory.

Within these categories, the scope can be narrowed using Language (i.e., what language does apply to), FormatProfile (e.g., SD, HD, UHD).

3.1.3 Referring to assets by language, track (ID), and description

AssetAvailability and AssetOrder refer to assets a little differently.

The following table shows the elements use to refer to tracks in Asset Availability and AssetOrder.

Element	AssetAvailability/Disposition	AssetOrder/Asset
Reference by language	Language	Language
Reference by specification	Track	TrackDescription
Reference by identifier		ID

The Language object is symmetrical. For example, both a studio and platform can refer to a "French SDH" or "English audio".

Note that AssetOrder includes identification while AssetAvailability does not. When a studio refers to a track, they include both description and identification. Without the description, the Platform would not know enough about the track. On the other hand, the Platform can refers identification provided in AssetOrder. Since the studio generated that AssetOrder, it knows what that ID means.

AssetOrder and AssetAvailability refer to tracks with some combination of the following:

- Language and track type

- ID – Referencing a track that was listed in AssetAvailability/Disposition/Track.
- Track Description – This is metadata descriptive of the track. This is used to ask for a track that has already been reference by in Disposition/Track. TrackDescription is symmetric to Disposition/Track, in that it can hold exactly the same data. However, there is no track ID.

Let's illustrate these.

3.1.3.1 Language

The first examples show referencing by language. This can appear in either or. The first example would be in AssetOrder/Asset/Language to express that that subtitles are required (or will be delivered).

```
<delivery:Language timedText="required”>fr-FR</delivery:Language>
```

This example, part of AssetAvailability/Disposition/Language, says that subtitles are available.

```
<delivery:Language timedText="available”>fr-FR</delivery:Language>
```

As you've probably noticed, the only difference is the value in the attribute that is consistent with context (ordering vs. availability).

Going a little deeper, consider a case where subs and dubs are both required.

```
<delivery:Language audio="required” timedText="required”>fr-FR</delivery:Language>
```

Note that there are other values than "required", such as "preferred" and "premium". To learn about these, see *Asset Ordering and Delivery*, Section 2.1.2.

3.1.3.2 Track Description

Track Description uses 'digital asset' metadata from Common Metadata. This is the same metadata structure found in Media Manifest/MMC/CPE Inventory.

Reference by characteristics is defined in Section 2.3.2.2.

Let's say we are referring to the French dub in AssetOrder. It would look like this:

```
<delivery:Description>
   <md:Audio>
      <md:Type>Primary</md:Type>
      <md:Language>fr-FR</md:Language>
   </md:Audio>
</delivery:Description>
```

Let's say we are referring to the French dub in AssetAvailability. It would look like this. Note the inclusion of the track ID. This track ID can be referenced later (i.e., in an AssetOrder or ProductStatus).

```
<delivery:Track>
   <md:Audio AudioTrackID="md:audiotrackid:eidr-x:AD07-310C-C59D-6785-C63A-G:audio.fr">
      <md:Type>Primary</md:Type>
      <md:Language>fr-FR</md:Language>
   </md:Audio>
</delivery:Track>
```

3.1.3.3 Identifier

Once a platform knows about a specific track, it can reference it by its identifier. This is the simplest and most reliable means to refer to tracks. Therefore, when track IDs are known, this is the preferred method to reference tracks.

The full description of using identifiers is in Section 2.3.1, but we'll provide some additional examples here.

The mechanism for referencing tracks by ID is AssetOrder/Asset/Reference/MDDFID. This structure supports all MDDF identifiers associated with assets, metadata and manifests.

In this example, we are referencing the track from the example in the previous section. Since the audio track ID was specified in the AssetAvailability record, it can simply be referenced. The fact it's a French dub is already known to both parties.

```
<delivery:Reference>
   <delivery:MDDFID>
      <delivery:AudioTrackID>md:audiotrackid:eidr-x:AD07-310C-C59D-6785-C63A-
G:audio.fr</delivery:AudioTrackID>
   </delivery:MDDFID>
</delivery:Reference>
```

As an order typically covers multiple tracks, as many tracks as necessary can be referenced in a single Reference object.

```
<delivery:Reference>
   <delivery:MDDFID>
      <delivery:AudioTrackID>md:audiotrackid:eidr-x:AD07-310C-C59D-6785-C63A-
G:audio.fr</delivery:AudioTrackID>
      <delivery:AudioTrackID>md:audiotrackid:eidr-x:AD07-310C-C59D-6785-C63A-
G:audio.en</delivery:AudioTrackID>
      <delivery:VideoTrackID>md:vidtrackid:eidr-x:AD07-310C-C59D-6785-C63A-G:video.main
</delivery:VideoTrackID>
      <delivery:SubtitleTrackID>md:subtrackid:eidr-x:AD07-310C-C59D-6785-C63A-G:subtitle.fr
/</delivery:SubtitleTrackID>
      <delivery:SubtitleTrackID>md:subtrackid:eidr-x:AD07-310C-C59D-6785-C63A-
G:subtitle.en</delivery:SubtitleTrackID>
   </delivery:MDDFID>
</delivery:Reference>
```

3.1.4 Note on some examples

Strictly for brevity, some examples below are not complete XML objects. The following shows the missing element structure. A full XML document would include the following with examples below replacing the ellipsis (…).

Asset Availability:

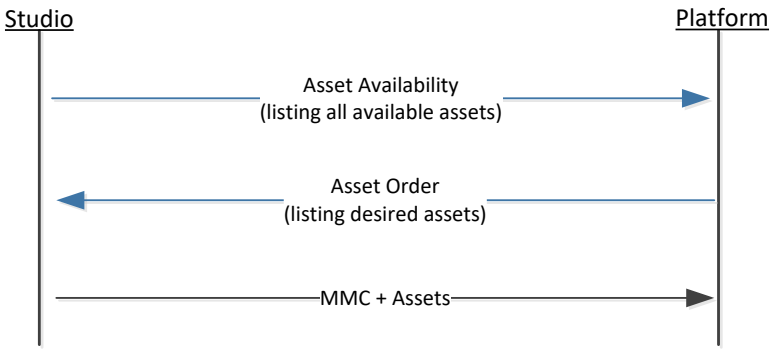
```
<?xml version="1.0" encoding="UTF-8"?>
<delivery:AssetAvailability xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:md="http://www.movielabs.com/schema/md/v2.8/md"
xmlns:manifest="http://www.movielabs.com/schema/manifest/v1.9/manifest"
xmlns:delivery="http://www.movielabs.com/schema/md/delivery/v1.0/delivery"
xsi:schemaLocation="http://www.movielabs.com/schema/md/delivery/v1.0/delivery delivery-v1.0.xsd">
   <delivery:Compatibility>
      <manifest:SpecVersion>1.0</manifest:SpecVersion>
   </delivery:Compatibility>
...
</delivery:AssetAvailability>
```

Asset Order:

```
<?xml version="1.0" encoding="UTF-8"?>
<delivery:AssetOrder xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:md="http://www.movielabs.com/schema/md/v2.8/md"
xmlns:manifest="http://www.movielabs.com/schema/manifest/v1.9/manifest"
xmlns:delivery="http://www.movielabs.com/schema/md/delivery/v1.0/delivery"
xsi:schemaLocation="http://www.movielabs.com/schema/md/delivery/v1.0/delivery delivery-v1.0.xsd">
   <delivery:Compatibility>
      <manifest:SpecVersion>1.0</manifest:SpecVersion>
   </delivery:Compatibility>
...
</delivery:AssetOrder>
```

3.2 Studio Push

For Push, the AssetAvailability object is used. When used in a pure Push model (i.e., no negotiation), the content provider is listing all the assets that are available. The presumption is that all assets are available now or will soon be available. A Studio Push would look something like this:



Note that if a platform is taking "everything", this process can be skipped. Media Manifest ([MMC]) has everything that's needed. So, in this context, a Push always has an acknowledgement of what assets are selected from the larger set.

The essence of the AssetRequest is the Disposition object. This is where the actual availability is listed. When we're doing a Push, assets already exist. These assets should be identified and described. This is done with the Track element as described in Section 3.1.3.2.

This simple example says that for the given ALID, there is one track. To add additional tracks, just add more Track instances. The content provider will likely provide more metadata, such encoding information and container information. MMC [MMC] provides explanation examples on encoding track (Inventory): www.movielabs.com/md/mmc

```
<delivery:AssetAvailability
...
   <delivery:Scope>
         <delivery:ALID>md:alid:eidr-s:AD07-310C-C59D-6785-C63A-G</delivery:ALID>
   </delivery:Scope>
   <delivery:AssetDispostion>
      <delivery:Track>
         <md:Audio AudioTrackID="md:audiotrackid:eidr-x:AD07-310C-C59D-6785-C63A-G:audio.fr">
            <md:Type>Primary</md:Type>
            <md:Language>fr-FR</md:Language>
         </md:Audio>
```

```
      </delivery:Track>
      <delivery:Track>
         <md:Audio AudioTrackID="md:audiotrackid:eidr-x:AD07-310C-C59D-6785-C63A-G:audio.en">
            <md:Type>Primary</md:Type>
            <md:Language>en</md:Language>
         </md:Audio>
      </delivery:Track>
      <delivery:StatusCode>available</delivery:StatusCode>
   </delivery:AssetDisposition>
</delivery:AssetAvailability>
```

As noted above, if no response is needed (i.e., platform is taking everything), it makes more sense just to use Media Manifest ([MMC]).

We're assuming the AssetAvailability was sent because an AssetOrder is expected. An Asset Order might look something like the following.

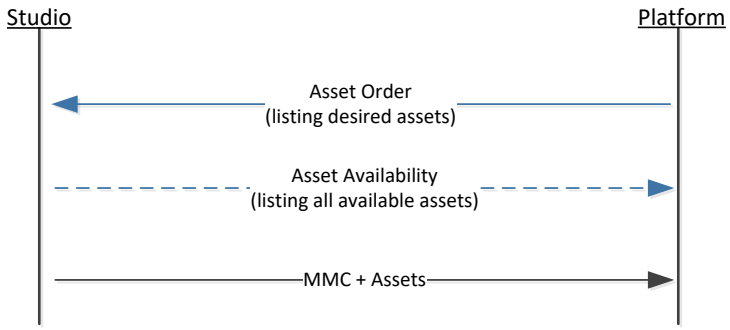
Note that the track IDs are known so these constitute the reference. In this example, only the French track is ordered.

Note also that the RequestCode is 'deliver', which is the appropriate value when the asset is known to exist.

```
<delivery:AssetOrder
...
   <delivery:Scope>
         <delivery:ALID>md:alid:eidr-s:AD07-310C-C59D-6785-C63A-G</delivery:ALID>
   </delivery:Scope>
   <delivery:Asset>
      <delivery:RequestCode>deliver</delivery:RequestCode>
      <delivery:Reference>
         <delivery:Description>
            <delivery:MDDFID>
               <delivery:AudioTrackID>md:audiotrackid:eidr-x:AD07-310C-C59D-6785-C63A-
G:audio.fr</delivery:AudioTrackID>
            </delivery:MDDFID>
         </delivery:Reference>
      </delivery:Asset>
</delivery:AssetOrder>
```

3.3 Platform Pull

The essential data in AssetOrder for a Pull are the RequestCode object and the Description object. A Platform Pull would look something like this:



In this use case, the RequestCode is 'request'. This code is valid whether or not the asset is known to exist.

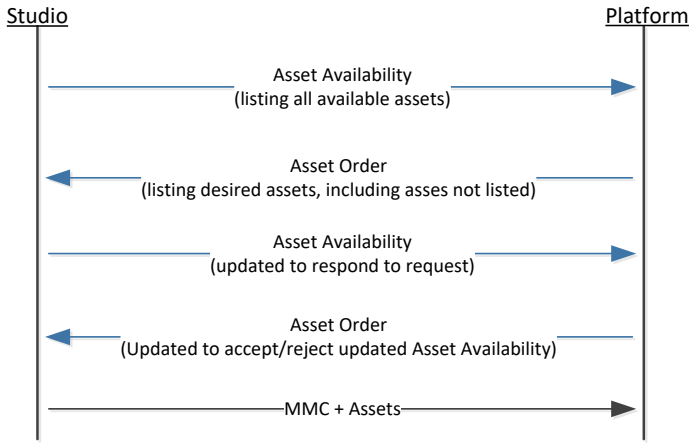
The AssetOrder/Description object is uses the same underlying types as AssetAvailability/Track so the encoding is very similar. So, you'll notice they are encoded almost identically. However, there is key difference in this Pull use case: The Platform has no knowledge of track identification which comes from AssetAvailability. Consequently, the Description object has no track IDs.

```
<delivery:AssetOrder
...
   <delivery:Scope>
         <delivery:ALID>md:alid:eidr-s:AD07-310C-C59D-6785-C63A-G</delivery:ALID>
   </delivery:Scope>
   <delivery:Asset>
      <delivery:RequestCode>request</delivery:RequestCode>
      <delivery:Description>
         <md:Audio>
            <md:Type>Primary</md:Type>
            <md:Language>fr-FR</md:Language>
         </md:Audio>
      </delivery:Description>
   </delivery:Asset>
</delivery:AssetOrder>
```

3.4 Negotiated exchange

As implied by the title, negotiation involves back and forth. The following example illustrates the back-and-forth that is possible.

In this example case, the studio initiated the exchange. The platform requests an additional track. And, it continues from there. It might look something like this:



A Studio-initiated exchange starts the same as Studio Push, using the example in that Section (0).

The retailer's response is an Order of existing assets. This is the same response as in Section 0, but we've added a twist. The Platform is requesting a German audio track.

RequestCode indicates which one is expected to be delivered, and which one is requested. Request dates are added (just to illustrate how these would appear).

```
<delivery:AssetOrder
...
```

```
   <delivery:Scope>
         <delivery:ALID>md:alid:eidr-s:AD07-310C-C59D-6785-C63A-G</delivery:ALID>
   </delivery:Scope>
   <delivery:Asset>
      <delivery:Reference>
         <delivery:RequestCode>deliver</delivery:RequestCode>
         <delivery:ExpectedDelivery>2019-09-12</delivery:ExpectedDelivery>
         <delivery:Description>
            <delivery:MDDFID>
               <delivery:AudioTrackID>md:audiotrackid:eidr-x:AD07-310C-C59D-6785-C63A-
G:audio.fr</delivery:AudioTrackID>
            </delivery:MDDFID>
         </delivery:Reference>
         <delivery:Description>
         <delivery:RequestCode>request</delivery:RequestCode>
         <delivery:ExpectedDelivery>2019-10-12</delivery:ExpectedDelivery>
            <md:Audio>
               <md:Type>Primary</md:Type>
               <md:Language>de</md:Language>
            </md:Audio>
         </delivery:Description>
      </delivery:Asset>
</delivery:AssetOrder>
```

The studio can respond with a revised AssetAvailability. The original AssetDisposition remains the same, but a new AssetDisposition is added for the new request.

There are two possible responses: rejected and accepted.

Note the StatusCode values. For French, status is now ‘processing’. For English, which was not requested, the status is still ‘available’. German was rejected.

```
<delivery:AssetAvailability
...
   <delivery:Scope>
         <delivery:ALID>md:alid:eidr-s:AD07-310C-C59D-6785-C63A-G</delivery:ALID>
   </delivery:Scope>
   <delivery:AssetDispostion>
      <delivery:Track>
         <md:Audio AudioTrackID="md:audiotrackid:eidr-x:AD07-310C-C59D-6785-C63A-G:audio.fr">
            <md:Type>Primary</md:Type>
            <md:Language>fr-FR</md:Language>
         </md:Audio>
      </delivery:Track>
      <delivery:StatusCode>processing</delivery:StatusCode>
   <delivery:AssetDispostion>
   </delivery:AssetDisposition>
      <delivery:Track>
         <md:Audio AudioTrackID="md:audiotrackid:eidr-x:AD07-310C-C59D-6785-C63A-G:audio.en">
            <md:Type>Primary</md:Type>
            <md:Language>en</md:Language>
         </md:Audio>
      </delivery:Track>
      <delivery:StatusCode>available</delivery:StatusCode>
   </delivery:AssetDisposition>
   <delivery:AssetDispostion>
      <delivery:Track>
         <md:Audio>
            <md:Type>Primary</md:Type>
            <md:Language>de</md:Language>
         </md:Audio>
      </delivery:Track>
      <delivery:StatusCode>rejected</delivery:StatusCode>
   </delivery:AssetDisposition>
</delivery:AssetAvailability>
```

If it was accepted, you might see something like the following. Note that 'available' StatusCode indicates it is available to be ordered, but has not yet been ordered because there are prices and dates to be determined. The delivery date is a month past the requested delivery date. And, there's a $200 price.

```
   <delivery:AssetDispostion>
      <delivery:Track>
         <md:Audio AudioTrackID="md:audiotrackid:eidr-x:AD07-310C-C59D-6785-C63A-G:audio.de">
            <md:Type>Primary</md:Type>
            <md:Language>de</md:Language>
         </md:Audio>
      </delivery:Track>
      <delivery:StatusCode>available</delivery:StatusCode>
      <delivery:ExpectedDelivery>2019-11-12</delivery:ExpectedDelivery>
      <delivery:BusinessTerms termName="price">
         <delivery:Money currency="usd">200</delivery:Money>
      </delivery:BusinessTerms>
   </delivery:AssetDisposition>
```

If the price is accepted, the platform responds with an AssetOrder. Note that the ExpectedDelivery has been updated and the RequestCode is 'deliver'.

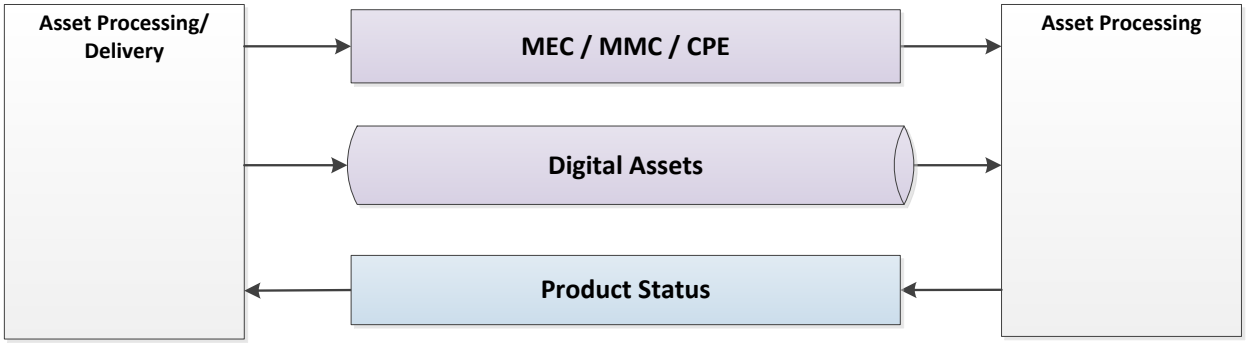
```
<delivery:AssetOrder
...
   <delivery:Scope>
         <delivery:ALID>md:alid:eidr-s:AD07-310C-C59D-6785-C63A-G</delivery:ALID>
   </delivery:Scope>
   <delivery:Asset>
      <delivery:Reference>
         <delivery:RequestCode>deliver</delivery:RequestCode>
         <delivery:ExpectedDelivery>2019-11-12</delivery:ExpectedDelivery>
         <delivery:Description>
            <delivery:MDDFID>
               <delivery:AudioTrackID>md:audiotrackid:eidr-x:AD07-310C-C59D-6785-C63A-
G:audio.de</delivery:AudioTrackID>
            </delivery:MDDFID>
         </delivery:Reference>
      </delivery:Asset>
</delivery:AssetOrder>
```

On the other hand, if the terms are not acceptable, the request can be rejected.

```
      <delivery:Reference>
         <delivery:RequestCode>cancel</delivery:RequestCode>
         <delivery:Description>
            <delivery:MDDFID>
               <delivery:AudioTrackID>md:audiotrackid:eidr-x:AD07-310C-C59D-6785-C63A-
G:audio.de</delivery:AudioTrackID>
            </delivery:MDDFID>
         </delivery:Reference>
```

4 PRODUCT STATUS

This core spec focuses on the objects shown in the illustration, although the QC Report structure is general enough to report on most anything (e.g., including Avails and Title List issues).



The term "Product" is used loosely here. It essentially means a collection of media, metadata and other data/files associated with some kind of offer. More specifically, it's anything in the purple boxes above (MEC, MMC, CPE, media, metadata, artwork, etc.).

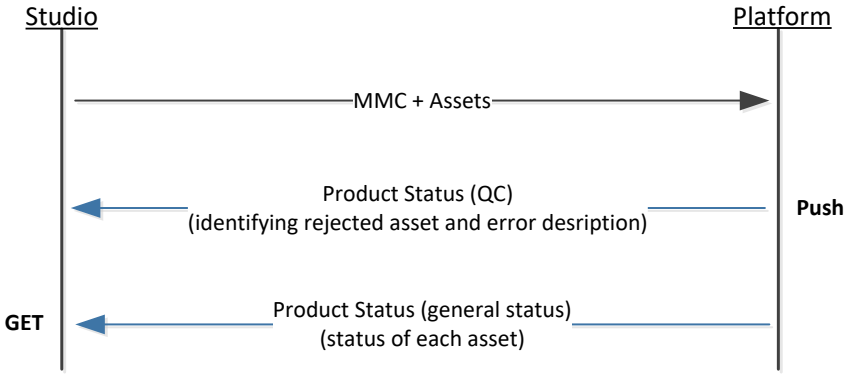
4.1 General information

4.1.1 ProductStatus vs. ObjectStatus (from Avails and Title List)

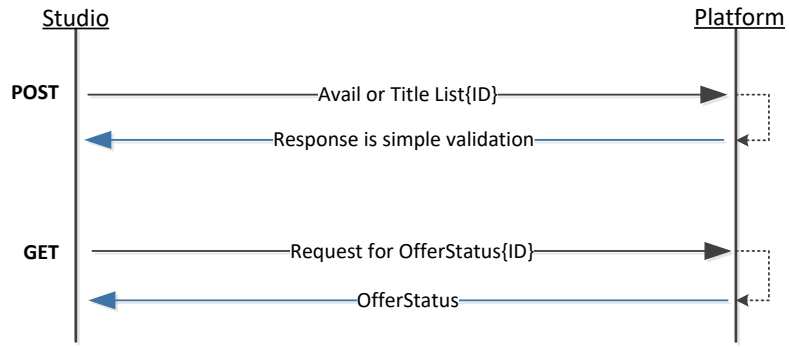
While the purpose of OfferStatus (found in the *Avails and Title List* spec) is to provide the business team with the status of a business offering, ProductStatus provides status to technical operations/fulfillment staff. There is some overlap because business people need to know if an offer is read to go, but detailed status is left to ProductStatus.

The primary difference between these elements is that AssetStatus reports detailed status (and errors) for objects that exist, while ObjectStatus provides higher-level status for objects that either already exist or are expected to exist.

Product Status might be used like this



In contrast, Offer Status would look like this



Let's say you've encountered an error in something delivered. By definition, it exists (i.e., the error was found in something that exists). And, you want to report an error. The correct element is AssetStatus. AssetStatus can also report progress on a specific asset (e.g., it is being processed, it is ready, it has been rejected). Progress codes are found in [delivery], Section 2.3.6.

Let's say an asset is expected but has not been delivered. Or, you want to report the status of every asset, whether it has been delivered or not. ObjectStatus should be used.

Because ObjectStatus refers to objects that either exist or not—and we wanted to keep it simple—there are only two attributes that can be used to describe the asset: Language and component. Component says what it is, either in general term (media, artwork, other), or specific terms (video, audio, timed text, descriptive, metadata, etc.). List of @component are in ProductProgess-type section (5.2.1) in [delivery].

4.1.2 Scope

In ProductStatus, Scope indicates the general context of the status. Guidance for encoding Scope can be found in Section 2.2.

However, since ProductStatus can refer to assets, the encoding described in Section 2.3.2.1 also applies.

4.1.3 Note on examples

Strictly for brevity, some examples below are not complete XML objects. The following shows the missing element structure. A full XML document would include the following with examples below replacing the ellipsis (…).

```
<?xml version="1.0" encoding="UTF-8"?>
<delivery:ProductStatus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:md="http://www.movielabs.com/schema/md/v2.8/md"
xmlns:manifest="http://www.movielabs.com/schema/manifest/v1.9/manifest"
xmlns:delivery="http://www.movielabs.com/schema/md/delivery/v1.0/delivery"
xsi:schemaLocation="http://www.movielabs.com/schema/md/delivery/v1.0/delivery delivery-v1.0.xsd">
   <delivery:Compatibility>
      <manifest:SpecVersion>1.0</manifest:SpecVersion>
   </delivery:Compatibility>
...

</delivery:ProductStatus>
```

4.2 Product Status

The general goal is to efficiently report status to determine if action needs to be taken.

ProductStatus is the root. Within ProductStatus, the key elements are Scope, OverallProgressCode and ObjectStatus.

4.2.1 What status is reported on

AssetStatus using ObjectStatus provides a general mechanism for reporting status. It is intended to provide high-level status on objects that already exist or are expected to exist. We expect that when using ObjectStatus it is reporting on a large set of assets, perhaps all assets associated with a title (via EIDR) or offer (via ALID).

The first division is around ObjectStatus/Category, with categories being things like 'feature', 'supplemental', 'promotional' (e.g., trailer) and 'image' (e.g., artwork). Within a Category, there can be multiple assets.

Let's say there are 100 assets (audio, video, timed text, metadata, artwork, etc.). Each asset gets a ObjectStatus/Progress instance. Progress/@language indicates the language of the asset (if applicable), and @component indicates the type of asset. As noted above, these can be either general (media, artwork, other) or specific (video, audio, timed text, descriptive, metadata, etc.).

4.2.2 ProgressCode

Progress Code is a concise means to report status. These are defined in [Delivery], Section 2.3.6. As noted in that section, acceptable values depend on whether it's referring to a single asset or multiple assets.

In some cases, ProgressCode stands alone and the asset is known by the context.

```
<delivery:OverallProgressCode>Missing</delivery:OverallProgressCode>
```

In other cases, specifically, where OverallProgressCode is used, one can specify a single overall progress code as well as progress codes for media, artwork, metadata, and other. Note that because metadata is missing, the overall progress code must be "Issues".

```
<delivery:OverallProgressCode Media="Ready" artwork="Ready"
metadata="Missing">Issue</delivery:OverallProgressCode>
```

In the example above, metadata was flagged as "missing". This is ok because generally there is only one metadata object. However, media and artwork can refer to multiple assets. Consequently, "Missing" or "Error" can only be used when it applies to all assets. It's generally best practice to just use a progress code of "Issue".

4.2.3 Reporting Overall Status

Overall status is reported using ProductStatus/OverallProgressCode. It is not necessary to use ObjectStatus or AssetStatus.

The first essential element is Scope. This identifies the scope of the status. ALID is used for deliveries associated with an ALID. EIDRURN is used for deliveries associated with a title (generally an edit). This example uses and ALID. And, we're going to constrain it further to be the assets associated with the US. Note that since we did not specify Language or FormatProfile, all languages and format profiles are covered.

The other necessary object is OverallProgressCode. Putting it together gives you this:

```
<delivery:Scope>
      <delivery:ALID>md:alid:eidr-s:AD07-310C-C59D-6785-C63A-G</delivery:ALID>
      <delivery:Region>
         <md:country>US</md:country>
      </delivery:Region>
</delivery:Scope>
<delivery:OverallProgressCode media="Ready" artwork="Ready"
metadata="Missing">Issue</delivery:OverallProgressCode>
```

This says that for the given ALID, in the US we're waiting for metadata.

4.2.4 Reporting Asset Status by category, language and component

Often it is important to report status more precisely. This is done with the ObjectStatus element.

Identification is the same as above. OverallProgressCode can be provided, although it is redundant. If included, it must be consistent with the individual asset statuses. For example, if an audio track is missing, media status must be "Missing" (i.e., waiting for track) or "Issue" (e.g., waiting for track, and it's late). Generally, status will be "Missing" immediately following a request.

ObjectStatus contains the asset status. Within ObjectStatus, the Progress object is required.

There are a variety of optional fields which might lead you to believe they are unnecessary. That is incorrect. Some of these fields must be provided, but which ones depends on what is being reported on.

Let's say status is for the feature's French subtitle.

```
<delivery:ObjectStatus>
   <delivery:Progress language="fr-FR" component="timed text">
      <delivery:ProgressCode>Missing</delivery:ProgressCode>
   </delivery:Progress>
</delivery:ObjectStatus>
```

Let's say the status is for a teaser, add Category as follows:

```
<delivery:ObjectStatus>
   <delivery:Category purpose="teaser">supplemental</delivery:Category>
   <delivery:Progress language="fr-FR" component="timed text">
      <delivery:ProgressCode>Missing</delivery:ProgressCode>
      <delivery:ExpectedDate>2019-09-10</delivery:ExpectedDate>
   </delivery:Progress>
</delivery:ObjectStatus>
```

Let's say you want to specify the date the asset is expected (typically by SLA)

```
<delivery:ObjectStatus>
```

```
   <delivery:Category purpose="teaser">supplemental</delivery:Category>
   <delivery:Progress language="fr-FR" component="timed text">
      <delivery:ProgressCode>Missing</delivery:ProgressCode>
      <delivery:ExpectedDate>2019-09-10</delivery:ExpectedDate>
   </delivery:Progress>
</delivery:ObjectStatus>
```

Multiple instances of Progress can be included to provide status on multiple objects within a category. Let's say we're also missing video

```
<delivery:ObjectStatus>
   <delivery:Category purpose="teaser">supplemental</delivery:Category>
   <delivery:Progress language="fr-FR" component="timed text">
      <delivery:ProgressCode>Missing</delivery:ProgressCode>
   </delivery:Progress>
   <delivery:Progress component="video">
      <delivery:ProgressCode>Missing</delivery:ProgressCode>
   </delivery:Progress>
</delivery:ObjectStatus>
```

Multiple instances of ObjectStatus can be included as well. In the following example, the feature is ready to go, but the teaser has a couple of issues.

```
<delivery:ObjectStatus>
   <delivery:Category >feature</delivery:Category>
   <delivery:Progress>
      <delivery:ProgressCode>Ready</delivery:ProgressCode>
   </delivery:Progress>
</delivery:ObjectStatus>
<delivery:ObjectStatus>
   <delivery:Category purpose="teaser">supplemental</delivery:Category>
   <delivery:Progress language="fr-FR" component="timed text">
      <delivery:ProgressCode>Missing</delivery:ProgressCode>
   </delivery:Progress>
   <delivery:Progress component="video">
      <delivery:ProgressCode>Missing</delivery:ProgressCode>
   </delivery:Progress>
</delivery:ObjectStatus>
```

4.2.5 Reporting Multiple Asset Status

If you need to get into details, use AssetStatus instead of ObjectStatus. AssetStatus allows you to reference assets individually.

This is exactly like a QC/Error Report described in the next section, but without ErrorDescription.

The following example says that an image with the given ID is missing.

```
<delivery:AssetStatus>
   <delivery:AssetReference>
      <delivery:MDDFID>
         <delivery:ImageID>md:imageid:eidr-x:AD07-310C-C59D-6785-C63A-G:hero1
         </delivery:ImageID>
      <delivery:MDDFID>
   </delivery:AssetReference>
   <delivery:ProgressCode>Missing</delivery:ProgressCode>
   <delivery:ErrorDescription>
</delivery:AssetStatus>
```

4.3 QC/Error Report

ProductStatus supports detailed error reporting. The structure of the response is dictated by what broke and how it broke.

Reporting the error itself (e.g., Video Hits, A/V Sync error, etc.) is relatively straightforward. Using QC Vocabulary found at www.movielabs.com/md/qcvocabulary, one can select the most appropriate term for the error. One has the option of adding description and specific parameters that expand upon the report (e.g., timecodes, regions of a picture, etc.).

Error reporting must precisely and unambiguously report *what* is broken so the studio or service provider knows what to fix. Section 0 provides detail on how reference objects.

What differentiates an error report from general status reporting is a ProductStatus/AssetStatus/ErrorDescription. This element defines the error, and what the error applies to.

4.3.1 Simple Error Report

A minimal AssetStatus contains the following

- AssetReference that refers to the object in error (e.g., an audio track).
- ProgressCode with a code indicating a problem with content. Generally, this is "Error".
- ErrorDescription including Error Category and Error Term based on the QC Vocabulary (referenced above).
    - Error category (ErrorDescription/ErrorCategory)
    - Error Term (ErrorDescription/ErrorTerm)

In this simple example, we have a bad MMC delivery. The error category is "DELIVERY-PACKAGE". The Term is "INVALID-MANIFEST". The Manifest ID is given.

```
<delivery:AssetStatus>
   <delivery:AssetReference>
      <delivery:MDDFID>
         <delivery:ManifestID>md:manifestid:eidr-s:AD07-310C-C59D-6785-C63A-G
         </delivery:ManifestID>
      <delivery:MDDFID>
   </delivery:AssetReference>
   <delivery:ProgressCode>Error</delivery:ProgressCode>
   <delivery:ErrorDescription>
      <delivery:ErrorCategory>DELIVERY-PACKAGE</delivery:ErrorCategory>
      <delivery:ErrorTerm>INVALID-MANIFEST</delivery:ErrorTerm>
   <delivery:ErrorDescription>
</delivery:AssetStatus>
```

That's all that is really needed.

4.3.2 Providing additional information in an Error Report

ErrorDescription contains CategorySpecificInfo to convey technical details around an error. Which child element is used depends on the error. For example,

CategorySpecificInfo/Audio is used for an audio error, and CategorySpecificInfo/Metadata is used to report a metadata problem.

The following illustrates technical data associated with artwork. There is a pixelization problem with the Hero 1 image in 100x100 area 300 pixels from the left and 400 pixels from the bottom.

```
<delivery:AssetStatus>
   <delivery:AssetReference>
      <delivery:MDDFID>
         <delivery:ImageID>md:imageid:eidr-x:AD07-310C-C59D-6785-C63A-G:hero1
         </delivery:ImageID>
      <delivery:MDDFID>
   </delivery:AssetReference>
   <delivery:ProgressCode>Error</delivery:ProgressCode>
   <delivery:ErrorDescription>
      <delivery:ErrorCategory>ARTWORK</delivery:ErrorCategory>
      <delivery:ErrorTerm>PIXELATION-BLURRY</delivery:ErrorTerm>
      <delivery:CategorySpecificInfo>
         <delivery:Artwork>
            <delivery:Area>
               <delivery:XOffset>300</delivery:XOffset>
               <delivery:YOffset>400</delivery:YOffset>
               <delivery:Width>100</delivery:Width>
               <delivery:Height>100</delivery:Height>
            </delivery:Area>
         </delivery:Artwork>
      </delivery:CategorySpecificInfo>
   <delivery:ErrorDescription>
</delivery:AssetStatus>
```