**Asset Ordering, Delivery and Tracking**

**Contents**

1 Introduction 1

1.1 Overview 1

1.2 Document Organization 2

1.3 Document Notation and Conventions 3

1.3.1 XML Conventions 3

1.3.2 General Notes 4

1.4 Normative References 5

1.5 Informative References 5

1.6 Best Practices for Maximum Compatibility 5

2 General Types Encoding 7

2.1 Attribute Groups 7

2.1.1 RangeAttributes 7

2.1.2 LanguageAssets-attr 7

2.2 Simple Types 9

2.3 Message and Terms Types 9

2.3.1 DeliveryPublisher-type and DeliveryPlatform-type 9

2.3.2 DeliveryHandling-type 10

2.3.3 DeliveryInstructions-type 11

2.3.4 DeliveryParams-type 11

2.3.5 DeliveryScope-type 12

2.3.6 Progress Codes, DeliveryProgressCode-type 13

2.4 Types that reference objects directly 14

2.4.1 DeliveryAssetReference-type 14

2.4.2 DeliveryImage-type 16

3 Asset Availability 18

3.1 AssetAvailability-type 18

3.1.1 AssetAvailabilityObject-type 19

4 Asset Order 21

4.1 AssetOrder-type 21

4.1.1 AssetOrderObject-type 21

4.1.2 AssetOrderTerms-type 22

5 Product Status 24

5.1 ProductObjectStatus 24

5.2 Product Object Status 25

5.2.1 ProductProgress-type 26

5.3 Product Asset Status 27

5.4 QC-specific Objects 27

5.4.1 QCErrorDescription-type 27

5.4.2 QCCategoryError-type 28

5.5 Logs 32

5.5.1 ProductLog-type 32

5.5.2 ProductLogEvent-type 32

6 NOTES 34


This work is licensed under a [Creative Commons Attribution 3.0 Unported License](http://creativecommons.org/licenses/by/3.0/).

**NOTE**: No effort is being made by the Motion Picture Laboratories to in any way obligate any market participant to adhere to Common Metadata. Whether to adopt the Common Metadata in whole or in part is left entirely to the individual discretion of individual market participants, using their own independent business judgment. Moreover, Motion Picture Laboratories disclaims any warranty or representation as to the suitability of the Common Metadata for any purpose, and any liability for any damages or other harm you may incur as a result of subscribing to this Common Metadata. **Revision History**

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Description** |
| 1.0 |  | Original Version |

# Introduction

This document defined data used in the delivery of assets, within the MovieLabs Digital Distribution Framework (MDDF). The following illustration shows the MDDF flow, with Asset Ordering and Delivery data shown in purple.



This specification is designed to work with other MDDF specifications or with proprietary/legacy specifications.

## Overview

The Asset Ordering and Delivery Process is addressed in three parts

* Rights Management – Generation and delivery of Avails or Title List and Offer Status
* Asset Planning – All processes associated with determining which assets (audio, video, subtitles, artwork, metadata, etc.) will be delivered
* Asset Delivery – Processes associated with the delivery and status of assets

These are illustrated in Figure 1 below.

The Rights Management process is covered by *Avails and Title List* and is not further discussed in this document. Offer Status is part of *Avails and Title List*. See [www.movielabs.com/md/avails](http://www.movielabs.com/md/avails) for more information.

Asset Planning determines what assets are delivered and when to meet obligations with partners. Asset policies are captured in “Content Delivery Requirements”. Avail or title-specific requests are included in Avail Confirmations, Asset Orders, and Asset Availability.

Asset Delivery has several parts including a Media Manifest Core (MMC) delivery spec, the assets themselves, and Product Status information including both general status of assets and error reporting. MMC is documented elsewhere ([www.movielabs.com/md/mmc](http://www.movielabs.com/md/mmc)), and this specification is neutral to assets delivered—we attempt to support almost any format. This specification documents Product Status.

Figure 1: Asset Distribution Workflow Composite



## Document Organization

This document is organized as follows:

1. Introduction—Provides background, scope and conventions
2. General Types Encoding
3. Profiles
4. Asset Planning and Delivery
5. Asset Order
6. Asset Availability
7. Product Status

## Document Notation and Conventions

As a general guideline, the key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119]. That is:

* “MUST”, “REQUIRED” or “SHALL”, mean that the definition is an absolute requirement of the specification.
* “MUST NOT” or “SHALL NOT” means that the definition is an absolute prohibition of the specification.
* “SHOULD” or “RECOMMENDED” mean that there may be valid reasons to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
* “SHOULD NOT” or “NOT RECOMMENDED” mean that there may be valid reasons when the particular behavior is acceptable, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
* “MAY” or “OPTIONAL” mean the item is truly optional, however a preferred implementation may be specified for OPTIONAL features to improve interoperability.

Terms defined to have a specific meaning within this specification will be capitalized, e.g. “Track”, and should be interpreted with their general meaning if not capitalized.

Normative key words are written in all caps, e.g. “SHALL”.

Normative requirements need not use the formal language above.

### XML Conventions

XML is used extensively in this document to describe data. It does not necessarily imply that actual data exchanged will be in XML. For example, JSON may be used equivalently.

This document uses tables to define XML structure. These tables may combine multiple elements and attributes in a single table. Although this does not align with schema structure, it is much more readable and hence easier to review and to implement.

Although the tables are less exact than XSD, the tables should not conflict with the schema. Such contradictions should be noted as errors and corrected.

#### Naming Conventions

This section describes naming conventions for Common Metadata XML attributes, element and other named entities. The conventions are as follows:

* Names use initial caps, as in InitialCaps.
* Elements begin with a capital letter, as in InitialCapitalElement.
* Attributes begin with a lowercase letter, as in initiaLowercaseAttribute.
* XML structures are formatted as Courier New, such as md:id-type
* Names of both simple and complex types are followed with “-type”

#### Structure of Element Table

Each section begins with an information introduction. For example, “The Bin Element describes the unique case information assigned to the notice.”

This is followed by a table with the following structure.

The headings are

* Element—the name of the element.
* Attribute—the name of the attribute
* Definition—a descriptive definition. The definition may define conditions of usage or other constraints.
* Value—the format of the attribute or element. Value may be an XML type (e.g., “string”) or a reference to another element description (e.g., “See Bar Element”). Annotations for limits or enumerations may be included (e.g.,” int [0..100]” to indicate an XML xs:int type with an accepted range from 1 to 100 inclusively)
* Card—cardinality of the element. If blank, then it is 1. Other typical values are 0..1 (optional), 1..n and 0..n.

The first row of the table after the header is the element being defined. This is immediately followed by attributes of this element, if any. Subsequent rows are child elements and their attributes. All child elements (i.e., those that are direct descendants) are included in the table. Simple child elements may be fully defined here (e.g., “Title”, “ ”, “Title of work”, “xs:string”), or described fully elsewhere (“POC”, “ ”, “Person to contact in case there is a problem”, “md:ContactInfo-type”). In this example, if POC was to be defined by a complex type defined as md:ContactInfo-type. Attributes immediately follow the containing element.

Accompanying the table is as much normative explanation as appropriate to fully define the element, and potentially examples for clarity. Examples and other informative descriptive text may follow. XML examples are included toward the end of the document and the referenced web sites.

### General Notes

All required elements and attributes must be included.

When enumerations are provided in the form ‘enumeration’, the quotation marks (‘’) should not be included.

UTF-8 [RFC3629] encoding shall be used when ISO/IEC 10646 (Universal Character Set) encoding is required.

## Normative References

[Avails] Content Availability Metadata, TR-META-AVAIL, <http://www.movielabs.com/md/avails>

[CM] Common Metadata, TR-META-CM, <http://www.movielabs.com/md/md>

[CMM] Common Media Manifest Metadata, TR-META-MMM, <http://www.movielabs.com/md/manifest>

[MEC] Media Entertainment Core, TR-META-MEC, , <http://www.movielabs.com/md/mec/>

[EIDR] Entertainment Identifier Registry (EIDR), <http://eidr.org/resources/>

[QCVocab] Quality Control (QC) Vocabulary, <http://www.movielabs.com/md/qcvocabulary>

[TR-META-CR] *Common Metadata Content Ratings*. [www.movielabs.com/md/ratings](http://www.movielabs.com/md/ratings). Note that a specific version is not referenced as it is intended that the latest version will be used. Referencing specifications may selection a specific version of the referenced document.

[TR-META-RS] Common Metadata Ratings Schema Definition, TR-META-RS, January 3, 2014, <http://www.movielabs.com/md/ratings/doc.html>

 [XML] “XML Schema Part 1: Structures”, Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn, W3C Recommendation 28 October 2004, <http://www.w3.org/TR/xmlschema-1/> and “XML Schema Part 2: Datatypes”, Paul Biron and Ashok Malhotra, W3C Recommendation 28 October 2004, http://www.w3.org/TR/xmlschema-2/

## Informative References

## Best Practices for Maximum Compatibility

Metadata typically evolves with the addition of new elements, attributes and vocabularies. Existing applications should be capable of accepting metadata, even though there might be more data than expected. Strict XML validation precludes an orderly evolution and can be counterproductive to the flexibility needed in real implementations.

Metadata specifications and schema updates are designed to support backwards compatibility. For example, element and attributes can be added, but required elements are not removed; or more generally ordinality of elements and attributes can be widened but not narrowed. Values are not changed in either syntax or semantics. Therefore, we strongly encourage implementations to either be diligent in tracking to the latest version, or follow the backwards compatibility rules provided here.

An XML document is considered compatible if its structure does not preclude the extraction of data from the document. For example, a document with additional elements and attributes do not preclude schema parsing and data extraction.

* Do not reject compatible XML documents, unless they fail schema validation against the definition for an exact version/namespace match.
* Extract data from compatible XML documents whenever possible
* It is allowable to ignore elements and attributes whose presence is not allowed in the specification and schema versions against which the implementation was built. For example, if the original schema allows one instance and three instances are found, the 2nd and 3rd instance may be ignored.

We will try to update metadata definitions such that following these rules work consistently over time. Sometimes, changes must be made that are not always backwards compatible, so we will do our best to note these.

# General Types Encoding

## Attribute Groups

### RangeAttributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
|  | **RangeAttributes-attr** |  |  |  |
|  | rangeCondition | Range Condition. See below. | xs:string | 0..1 |
|  | rangeRank | Relative ranking within equal rangteCondition, or if rangeCondition is unspecified. 0 is highest rank. | xs:nonNegativeInteger | 0..1 |

RangeCondition defines the range of acceptable technical parameters. RangeCondition is an xs:string and typically an attribute (@rangeCondition).

When values are expressed,

Acceptable values for @rangeCondition are as follows

* ‘min’ – Represents minimum requirement. If numeric, lower values are not accepted.
* ‘max’ – Represents the maximum acceptable value. If numeric, higher values are not accepted.
* ‘preferred’ – Represents preferred condition or value.
* ‘acceptable’ – Represents a condition or value that is acceptable but not desired. There may be negative consequences of using this condition, such as lower quality.

### LanguageAssets-attr

The LanguageAssets attribute group defines assets associated with a language. It is used both to define rules and to reference assets.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Group** | **Attribute** | **Definition** | **Value** | **Card.** |
| **DeliveryLanguageRules-attr** |  |  |  |  |
|  | audio | Audio in this language is required or desired.  | xs:string | 0..1 |
|  | video | Video in this language is required or desired. Only applies if there are multiple versions of the video. | xs:string | 0..1 |
|  | timedText | Timed text localization requirements as specified below | xs:string | 0..1 |
|  | SDH | SDH Timed text localization requirements as specified below | xs:string | 0..1 |
|  | descriptive | Descriptive audio is required or desired. See encoding information below. | xs:string | 0..1 |
|  | signed | Video with signing is required or desired. See encoding information below. | xs:string | 0..1 |
|  | metadata | Localized metadata is required or desired. See encoding information below. | xs:string | 0..1 |
|  | subdubPreferred | Indicates that timed text or dub is preferred. | xs:string | 0..1 |

The following values apply to all attribute. They are used to indicate the disposition of an asset.

* ‘available’ – The asset is available, or will be within the terms of an agreement
* ‘offered’ – The asset can be made available (e.g., can be requested or can be ordered) ‘available’ – Asset is available, but has not been requested
* ‘processing’ – Asset is being processed for delivery
* ‘delivered’ – Asset has been delivered and considered completed unless recipient indicates otherwise
* ‘rejected’ – Asset has been requested, but will not be delivered

The following values apply specific attributes. They are used to specify requirement for assets.

@audio is encoded as follows:

* ‘required’ – Localized audio is required. Can be delivered in any format as opposed to ‘premium’ where premium formats are required. Default for Original.
* ‘premium –Localized asset is required in premium format (i.e., multichannel or object-based audio).
* ‘desired – Localized audio is desired. It is not a requirement for launch.

@timedText and @SDH are encoded as follows:

* ‘required’ –Timed text is required.
* ‘desired –Timed text is desired. It is not a requirement for launch.
* ‘either—Either language or ‘SDH’ subtitles are required. Both @timedText and @SDH must be encoded ‘either’.
* Note: If both language and ‘SDH’ subtitles are required. Both @timedText and @SDH must be encoded ‘required’

@descriptive and @signed is encoded as follows:

* ‘required’ – Localized asset is required.
* ‘preferred’ – Localized asset is desired. It is not a requirement for launch.

@subdubPreferred is used to indicated that either timed text or audio is required, and which one is preferred. When used, @audio must be ‘required’ or ‘premium’; and at least one of @timedText and @SDH must be ‘required’ or both @timedText and @SDH must be ‘either’.

* ‘sub’ – Indicates timed text is preferred
* ‘dub’ – Indicates audio dub is preferred

## Simple Types

Currently, there are no Simple Types in this schema.

## Message and Terms Types

### DeliveryPublisher-type and DeliveryPlatform-type

These fields are provided to allow the recipient of a message to see who it is from, and who it is for; especially, when those parties are ambiguous.

There are up to three parties involved in each transaction: Content Provider/Publisher/Studio, Platform/Retailer and Service Provider. Information might be exchanged between studios and platforms directly (in either direction), or via service providers.

DeliveryPublisher-type and DeliveryPlatform-type provides information about who is sending or receiving information. Whether the Publisher or Platform is the sender or receiver depends on the direction of the message. DeliveryPublisher-type is used to define the Publishers and/or Service Providers acting on behalf of Publishers, whether it is the sender or recipient. DeliveryPlatform-type provides the same data for Platforms and their Service Providers.

A source or destination can have multiple Publisher or Platform instances. This allows a single transaction to apply to a variety of parties. For example, given a company organized around territorial business units (e.g., “Sofaspud Films, US; and Sofaspud Films, EMEA), multiple instances can indicate that this transaction applies to multiple business units.

When service providers are in the transaction, from the standpoint of these interfaces, they are a proxy for another party. For example, a service provider might send information to a platform on behalf of a studio; or, a platform might send information to a service provider for eventual delivery to a studio.

ServiceProvider should only be included when the Service Provider is then sender or recipient of the message. Service Providers are assumed to be single entities, so there is no need for multiple instances.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **DeliveryPublisher-type** |  |  |  |  |
| Publisher |  | Publisher for whom the document was created | md:OrgName-type | 0..n |
| ServiceProvider |  | Service Provider delivering document | md:OrgName-type | 0..1 |
| Contact |  | Contact information for this document, typically from a Service Provider. | md:ContactInfo-type | 0..1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **DeliveryPlatform-type** |  |  |  |  |
| Platform |  | Platform/Retailer for whom the document was created | md:OrgName-type | 0..n |
| ServiceProvider |  | Service Provider delivering document | md:OrgName-type | 0..1 |
| Contact |  | Contact information for this document, typically from a Service Provider. | md:ContactInfo-type | 0..1 |

### DeliveryHandling-type

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **DeliveryHandling-type** |  |  |  |  |
| Comments |  | Any comments. Should be included if ExceptionFlag=’true’ | xs:string | 0..1 |
| ExceptionFlag |  | Indicates message requires human attention | xs:boolean | 0..1 |
| ResponseDate |  | Expected response date | xs:date | 0..1 |
|  | dateIsTarget | If ‘true’ indicates ResponseDate is not a hard deadline. Details determined bilaterally. | xs:boolean | 0..1 |

### DeliveryInstructions-type

DeliveryInstructions-type extends DeliveryHandling-type to include OrderID. This is for cases where an order applies. Note that not all uses of DeliveryHandling-type apply to an Order (e.g., Avails-related requests).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **DeliveryInstructions-type** |  |  | delivery:DeliveryHandling-type |  |
| OrderID |  | Order identifier | md:id-type | 0.1 |

### DeliveryParams-type

DeliveryParams-type includes delivery parameters that are common across media types, metadata, promotional, supplemental and other materials.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **DeliveryParams-type** |  |  |  |  |
| LeadTime |  | Lead time for deliverables relative to window start date. Negative values represent time before window. | xs:duration | 0..1 |
|  | durationIsTarget | If ‘true’ LeadTime is a target; that is, not a fixed duration | xs:boolean | 0..1 |
| Priority |  | Priority of request. Lower number is higher priority, with 0 being the highest. | xs:nonNegativeInteger | 0..1 |
| AdditionalInstructions |  | Any additional instructions | xs:string | 0..1 |
| Terms |  | Any additional terms | md:Terms-type | 0..n |

LeadTime is expressed as a negative duration for deliverables that occur prior to the window (the typical case).

durationIsTarget indicate that LeadTime are aspirational. The degree to which this must be honored is subject to bilateral service level agreements.

Priority is specified relative to a given DueDate. Note that processing of Priority will require Best Practices that define factors to optimize when prioritizing deliveries of different types across different dates (i.e., factoring in urgency versus importance).

### DeliveryScope-type

Delivery Scope allows an object such as an asset order or status report to define the context for the message. That is, defining the scope of the assets for which the object was generated. For example, if the delivery is associated with a particular Avail in France, one would use the ALID and Region to refer to the delivery. If the data is specific to a language or format profile, the Language and FormatProfile elements can be used. TransactionID (same as AvailID in Excel) is an efficient means of referring to a specific Avail over (Transaction element in XML, or row in Excel).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **DeliveryScope-type** |  |  |  |  |
| ALID |  | ALID | md:id-type | 0..1 |
| AlternateID |  | Alternate ID from Avail | md:id-type | 0..n |
| TransactionID |  | Transaction ID from Avail | md:id-type | 0..n |
| EIDRURN |  | EIDR in URN format | md:EIDRURN-type | 0..1 |
| Region |  | Region(s) | md:Region-type | 1..n | 0..1 choice |
| ExcludedRegion |  | Excluded Region(s) | md:Region-type | 0..n |
| Language |  | Language | xs:language | 0..n |
|  | asset | Corresponds with LocalizationOffering in Avails [Avails], Section 2.2.2.1 (i.e., ‘sub’, ‘dub’, ‘subdub’, ‘any’) | xs:string | 0..1 |
| FormatProfile |  | Format Profile as defined in Avails [Avails], Section 2.2.3 | xs:string | 0..n |
|  | HDR | xs:string |  |
|  | WCG | xs:string |  |
|  | HFR | xs:string |  |
|  | NGAudio | xs:string |  |

### Progress Codes, DeliveryProgressCode-type

Progress codes provide general guidance regarding the status of a delivery. Specific information is found in ErrorDescription, when included.

Depending on context, progress codes may refer to specific assets or to multiple assets.

When referring to a single asset, Progress Code values include

* ‘Ready’ – Asset has been delivered and approved. No additional delivery is required.
* ‘In-Process’ – There is no status to report as asset is being processed
* ‘Missing’ – Asset is expected but has not been delivered.
* ‘Error’ – There is an issue with the asset.

Each asset has a state, but when referring multiple assets, the status could be a combination of codes (i.e., some might be *ready*, some might be *in-process*, some might be *missing*, and some might have *errors*). Consequently, Progress Code values for multiple assets are defined as follows:

* ‘Ready’ – All assets are ready.
* ‘In-Process’ – There is no status to report as assets are being processed
* ‘Issue’ – One or more assets are missing or in error. If there are multiple issues, there can be an ErrorDescription instance for each issue.

#### The DeliveryProgressCode-type

The DeliveryProgressCode-type complex type is used when referring to assets that have some combination of audiovisual media, artwork and metadata. It allows progress to be reported against each.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **DeliveryProgressCode-type** |  | Progress code | xs:string (by extension) |  |
|  | media | Progress code for media part | xs:string | 0..1 |
|  | artwork | Progress code for image/artwork part | xs:string | 0..1 |
|  | metadata | Progress code for metadata part | xs:string | 0..1 |
|  | other | Progress code for other parts (e.g., interactive) | xs:string | 0..1 |

## Types that reference objects directly

### DeliveryAssetReference-type

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **DeliveryAssetReference-type** |  |  |  | 0..1 |
| TrackReference |  | TrackReference per [Manifest], Section 2.2.3 | xs:string | 0..n |
| TrackIdentifier |  | TrackIdentifier per [Manifest], Section 2.2.3 | md:ContentIdentifier-type | 0..n |
| EIDRURN |  | EIDR identifier along with scope/structural type | md:EIDRURN-type | 0..n |
| ManifestID |  | Reference track identifiers as per [Manifest] | delivery:DeliveryMDDFID | 0..n |
| FileInfo |  | Reference to a file | manifest:FileInfo-type | 0..n |
| Container |  | Reference to a container | Manifest:ContainerInfo-typeReference-type | 0..n |
| IMFRef |  | Reference to information in an Interoperable Master Format (IMF) file. | Delivery:DeliveryIMF-type | 0..n |
| OtherIdentifier |  | Any other applicable identifier | md:ContentIdentifier-type | 0..n |

#### DeliveryMDDFID-type

Allows reference via MDDF identifiers. This includes ContentID for metadata and various identifiers used in Media Manifest [manifest].

The first section is track IDs. Then it gets into other Manifest objects such as Presentations.

When using something other than MDDF, use TrackIdentifier for track, and OtherIdentifier for other objects.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **DeliveryMDDFID-type** |  |  |  |  |
| AudioTrackID |  | Audio track ID | manifest:AudioTrackID-type | (choice) |
| VideoTrackID |  | Video track ID | manifest:VideoTrackID-type |
| SubtitleTrackID |  | SubtitleTrack ID | manifest:SubtitleTrackID-type |
| ImageID |  | Image ID | manifest:ImageTrackID-type |
| InteractiveTrackID |  | Interactive object (e.g., app) ID | manifest:InteractiveTrackID-type |
| ContentID |  | Content ID. Content ID references metadata, so this ID is used to reference a Common Metadata/MEC metadata object. | md:ContentIID-type |
| AncillaryTrackID |  | Ancillary track ID | manifest:AncillaryTrackID-type |
| TextObjectID |  | Text object ID | manifest:TextObjectTrackID-type |
| PresentationID |  | Presentation ID | manifest:PresentationID-type |
| PlayableSequenceID |  | Playable Sequence ID | manifest:PlayableSequence-type |
| PictureGroupID |  | Picture Group ID | manifest:PictureGroup-type |
| AppGroupID |  | Application Group ID | manifest:AppGroupID-type |
| TextGroupID |  | Text Group ID | manifest:TextGroupID-type |
| ExperienceID |  | Experience ID | manifest:ExperienceID-type |
| TimedSequenceID |  | Timed Sequence ID | manifest:TimedSequenceID-type |
| TransactionID |  | Avails and Title List Transaction ID | md:id-type |

#### DeliveryIMFRef-type

References UUIDs for IMF CPLs, OPLs and virtual tracks.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **DeliveryIMFRef-type** |  |  | extension of manifest:PresentationIMFRef-type |  |
| **VirtualTrackID** |  | Referenced virtual track ID(s) | xs:string | 0..n |

NOTE: This object may need to be extended to reference other components of an IMF, particularly individual files. This specificity might be needed to more granularly request components or to report errors with more specificity.

### DeliveryImage-type

This object defines image technical characteristics. A set of image characteristics is called an Image Profile.

References to Common Metadata types in this section refer to object in DigitalAssetImageData-type, as defined in [CM] section 5.2.8, with the same name. Pixels are assumed to be square.

The image profile may be given a name in @imageProfileName. If this name is absent, it is assumed that all images will conform to this profile. Otherwise, artwork definitions must reference a named profile.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **DeliveryImage-type** |  | Base type for this element is standard delivery parameters defined in DeliveryParams-type. | delivery:DeliveryParams-type (by extension) |  |
|  | imageTechProfileName | Unique name of technical profile. If there is only one profile of this type and @default=‘true’, this need not be included. | md:id-type | 0..1 |
|  | default | Is this the default profile. If ‘true’, it is. If absent or ‘false’ it is not default. At most one instance can be the default | xs:boolean | 0..1 |
|  | purpose | Purpose of image | xs:string | 0..1 |
| Encoding |  | As per Common Metadata definition. One for each acceptable encoding method. | xs:string | 0..n |
| AlphaAllowed |  | Is alpha channel supported (i.e., transparency). ‘true’ means yes. This must be absent or ‘false’ for encoding types that do not support alpha. | xs:boolean | 0..1 |
| DynamicRangeProfile |  | As defined in [CM] | xs:string | 0..1 |
| ColorGamutProfile |  | As defined in [CM] | xs:string | 0..1 |
| Compliance |  | As defined in [CM] | md:Compliance-type | 0..1 |
| MaxFileSize |  | Maximum file size in bytes for file of this type | xs:nonNegativeInteger | 0..1 |
| Term |  | Additional terms that apply to this Profile | md:Terms-type | 0..n |

# Asset Availability

The Asset Availability describes the status of asset delivery from the studio to the retailer. This can include assets in any stage of delivery. Some conditions include

* Assets that have been delivered (retailer perspective on that delivery notwithstanding)
* Assets that are being prepared
* Assets that could potentially be provided by request (perhaps with a fee)

Note that asset status information is sent in both directions the mirror image of this object is AssetAvailability-type sent from the retailer to the studio.

## AssetAvailability-type

AssetAvailability-type describes the state of a particular localization or track.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **AssetAvailability-type** |  |  |  |  |
|  | updateNum, workflow, updateDeliveryType, versionDescription, timestamp | Workflow attributes | md:Worflow-attr | 0..1 |
| Compatibility |  | Spec compatibility | md:Compatibility-type |  |
| Source |  | Source of this request | delivery:DeliveryPublisher-type | 0..1 |
| Destination |  | Platform or service provider receiving status  | delivery:DeliveryPlatform-type | 0..1 |
| DeliveryID |  |  | md:id-type | 0..1 |
| Description |  | Description of request | xs:string | 0..1 |
| Scope |  | Information to associate the order with the offer associated with this delivery. | delivery:DeliveryScope-type | 0..1 |
| AssetDisposition |  | Status of asset or group of assets | delivery:AssetAvailabilityObject-type | 0..n |
| Instructions |  | Any other instructions | xs:string | 0..1 |

### AssetAvailabilityObject-type

This complex type contains the disposition of an ‘object’ which is either assets associated with a language (e.g., French subtitle track) or a specific asset as it would be described in Media Manifest Inventory.

If the content provider is expressing asset disposition from the perspective of languages, AssetLanguage is used. This construct defines a set of assets associated with a particular language. The status applies to all objects referenced within the AsssetLanguage object. For example, if @audio and @SDH are set, then those assets are being statused. @subdub indicates subtitles and/or dubs are provided at the discretion of the content provider.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **AssetAvailabilityObject-type** |  |  |  |  |
| Language |  | Language of set of assets for which disposition is provided. | xs:language |  | choice |
|  | audio, timedText, SDH, etc. |  | delivery:LanguageAssets-attr | 0..1 |  |
|  | subdub | If true, audio dubs, timed text, or both are provided at the discretion of the content provider. Does not apply to original language. | xs:boolean | 0..1 |  |
|  | OV | Audio is the Original Version (original language) | xs:boolean | 0..1 |  |
| Track |  | Describes a single track as it would be described in Media Manifest.  | Manifest:InventorySingleTrack-type |  |
| StatusCode |  | Code that indicates status of asset or assets identified in ObjectReference or ObjectDescription | xs:string |  |
| ErrorReference |  | ErrorReference associated with ErrorDescription in ProductStatus. Associated with rework (i.e., StatusCode=’rework’).  | xs:string | 0..n |
| ExpectedDelivery |  | Expected delivery date | md:YearDateOrTime-type | 0..1 |
| BusinessTerms |  | Business terms, such as cost to generate or deliver asset | md:Terms-type | 0..n |
| TechnicalTerms |  | Additional technical terms relating to asset delivery | md:Terms-type | 0..n |
| Instructions |  | Any other instructions | xs:string | 0..1 |

StatusCode indicates the status of the particular asset. Values include

* ‘available’ – Asset is available, but has not been requested
* ‘processing’ – Asset is being processed for delivery
* ‘delivered’ – Asset has been delivered and considered completed unless recipient indicates otherwise
* ‘rework’ – Being reworked following an QC report
* ‘rejected’ – Asset has been requested, but will not be delivered
* ‘recalled’ – Asset has been delivered, but has a problem and should not be used

Language attributes values are defined in LanguageAsset-attr in Section 2.1.2. The value ‘available’ should be used when the asset is available. ‘offered’ should be used when the asset can be made available (e.g., can be requested or can be ordered). When an asset is available to order, one might use business terms to dictate the terms.

# Asset Order

An Asset Order defines objects to be delivered.

## AssetOrder-type

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **AssetOrder-type** |  |  |  |  |
|  | updateNum, workflow, updateDeliveryType, versionDescription, timestamp | Workflow attributes | md:Worflow-attr | 0..1 |
| Compatibility |  | Spec compatibility | md:Compatibility-type |  |
| Source |  | Source of this message | delivery:DeliveryPlatform-type | 0..1 |
| Destination |  | Publisher to whom the status is being sent | delivery:DeliveryPublisher-type | 0..1 |
| DeliveryID |  |  | md:id-type | 0..1 |
| Description |  | Description of request | xs:string | 0..1 |
| Scope |  | Information to associate the order with the offer associated with this delivery. | delivery:DeliveryScope-type |  |
| Asset |  | Identifies assets and specifies terms specific to that asset | delivery:AssetOrderObject-type | 0..n |
| TermsAcrossAssets |  | Specifies terms that apply to all assets identified in the Asset object | delivery:AssetOrderTerms-type | 0..n |
| Instructions |  | Any other instructions | xs:string | 0..1 |

### AssetOrderObject-type

AssetOrderObject-type specifies the object to be delivered, and possibly terms specific to that object.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **AssetOrderObject-type** |  |  | Delivery:AssetOrderTerms-type (by extension) |  |
| Language |  | Order assets based on language | xs:language |  | (choice)0..n |
|  | audio, timedText, SDH, etc. |  | delivery:LanguageAssets-attr | 0..1 |
| OV |  | Order assets based on the Original Version (original language).  | xs:language |  |
|  | audio, timedText, SDH, etc. |  | delivery:LanguageAssets-attr | 0..1 |
| TrackDescription |  | Reference to objects, such as tracks, by description (e.g., *French dub*). | manifest:Inventory-type |  |
| ID |  | Reference to objects such as tracks, requested  | delivery:DeliveryAssetReference-type |

### AssetOrderTerms-type

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **AssetOrderTerms-type** |  |  |  |  |
| RequestCode |  | Code that indicates order status for the object | xs:string |  |
| ExpectedDelivery |  | Expected delivery date | md:YearDateOrTime-type | 0..1 |
| BusinessTerms |  | Business terms, such as cost to generate or deliver asset | md:Terms-type | 0..n |
| TechnicalTerms |  | Additional technical terms relating to asset delivery | md:Terms-type | 0..n |
| Instructions |  | Any other instructions | xs:string | 0..1 |

RequestCode indicates how the request should be handled. For example, it could be a request that assets be delivered, it could be a request of estimated delivery, or it could be a request to price the delivery of assets. RequestCode applies to the entire Scope. For example, if Scope is Region/Country=“de”, and RequestCode is ‘deliver’, the request is to deliver everything for Germany.

Values for RequestCode include

* ‘deliver’ – Deliver asset
* ‘redeliver’ –There was a problem with delivery and redelivery is requested
* ‘cancel’ – Asset is not needed. Request can be cancelled.
* ‘request’ – Asset is not on an AssetAvailability list, but is requested to be delivered.

# Product Status

Product Status provides the means for communicating status once there is some agreement on the assets to be delivered.

Depending how ProductStatus is used in the workflow, assets are referenced by description (e.g., “The French dub”), or precisely (e.g., “Track with track ID = …”). The former case usually applies before specific assets are known; either before delivery, or a byproduct of an error (e.g., missing asset). The latter when specific assets are being referenced, such reporting a QC issue with a particular track. The descriptive reference is implemented in the ObjectStatus portion of this element. The precise references are implemented in AssetStatus.

A Quality Control (QC) reports is a special case of a ProductStatus object. This report provides the means to identify issues media, metadata and other files. In the simplest form, the QC Report can identify the object in question and the convey associated issue. The QC Report also supports additional data associated with particular media types. For example, timecode ranges can be conveyed for any audio, video and timed text. For uniformity, errors are reported using the standardized QC Vocabulary found in [QCVocab]. What distinguishes a QC report is the presence of ErrorDescription, an object that provides specific information about anomalies associated with deliveries.

## ProductObjectStatus

ProductStatus-type is the defines the ProductStatus element.

This element provides two means of reporting status, reflected in ObjectStatus and AssetStatus. The primary between these elements is AssetStatus reports detailed status (and errors) for objects that exist, while ObjectStatus provides high-level status for objects that either already exist or are expected to exist.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **ProductObjectStatus-type** |  |  |  |  |
|  | udpateNum, workflow, updateDeliveryType, versionDescription | Common set of workflow attributes (defined in Common Metadata) | md:Workflow-attr |  |
| Compatibility |  | Spec compatibility | md:Compatibility-type |  |
| Source |  | Source of this message | delivery:DeliveryPlatform-type | 0..1 |
| Destination |  | Publisher to whom the status is being sent | delivery:DeliveryPublisher-type | 0..1 |
| DeliveryID |  | ID associated with the delivery | md:id-type | 0..1 |
| Description |  | Description of status (overview) | xs:string | 0..1 |
| Scope |  | Information to tie this status to an Avail or other offer | delivery:DeliveryScope-type | 0..1 |
| OverallProgressCode |  | Overall status progress code(s) | Delivery:DeliveryProgressCode-type | 0..1 |
| ObjectStatus |  | Status of a category of assets referenced descriptively.  | delivery:ProductObjectStatus-type | 1..n | 0..1 choice |
| AssetStatus |  | Status of specific assets referenced by identifiers or names.  | delivery:ProductAssetStatus-type | 1..n |
| Instructions |  | Handling instructions. Includes exception flag.  | delivery:Instructions-type | 0..1 |
| Log |  | Event Log | delivery:ProductLog-type | 0..1 |

## Product Object Status

ProductAssetStatus-type provides status of asset delivery processing.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **ProductAssetStatus-type** |  |  |  |  |
| Category |  | Category of object. | xs:string | 0..n |
|  | purpose | Purpose of object within category | xs:string | 0..1 |
| AssetLanguage |  | Reference to Asset | delivery:DeliveryAssetReference-type | 0..n |
| Progress |  | Progress of assets | delivery:ProductProgress-type | 0..n |
| Comments |  | Any additional comments | xs:string | 0..1 |
| Log |  | Log of previous events | delivery:DeliveryLogEvent-type | 0..1 |
| Instructions |  | Handling instructions. Includes exception flag.  | delivery:Instructions-type | 0..1 |

Category is encoded as follows:

* ‘feature’ – Object is feature
* ‘supplemental’ – Object is supplemental material (e.g., Extras/Bonus/VAM)
* ‘promotional’ – Object is promotional material, typically a trailer
* ‘image’ – Object is an image, typically artwork

### ProductProgress-type

ProductProgress-type defines progress with varying precision.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **ProductProgress-type** |  |  |  |  |
|  | language | Language associated with progress. If absent, progress applies to all languages | xs:string | 0..1 |
|  | component | Asset component type. If absent, progress applies to all components. See below | xs:string | 0..1 |
| ProgressCode |  | Associated progress code. See Section 2.3.6 | xs:string |  |
| ExpectedDate |  | Date when asset is (or was) expected | xs;date | 0..1 |

@component is defined as follows:

* General categories
	+ ‘media’ – media including audio, video and timed text
	+ ‘artwork’ – Artwork; typically metadata artwork
	+ ‘other’ – Anything not covered by another category
* Specific categories
	+ ‘video’ – video track(s)
	+ ‘audio’ – audio track(s). Can be original or dub depending on language.
	+ ‘timed text’ – timed text/subtitles
	+ ‘descriptive’ – Descriptive audio
	+ ‘metadata’ – Metadata

## Product Asset Status

ProductAssetStatus-type provides status of asset delivery processing.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **ProductAssetStatus-type** |  |  |  |  |
| AssetReference |  | Reference to Asset | delivery:DeliveryAssetReference-type | 0..n |
| ProgressCode |  | Progress Code. See Section 2.3.6 | xs:string |  |
| ErrorDescription |  | Description of error associated with progress | delivery:QCErrorDescription-type | 0..1 |
| Comments |  | Any additional comments | xs:string | 0..1 |
| Log |  | Log of previous events | delivery:DeliveryLogEvent-type | 0..1 |
| Instructions |  | Handling instructions. Includes exception flag.  | delivery:Instructions-type | 0..1 |

## QC-specific Objects

### QCErrorDescription-type

QCError-Desription-type provide information about the error. ErrorCategory and ErrorTerm are from QC Vocabulary [QCVocab].

In the form of CategorySpecific details on the specific error can be provided. For example, in anything time-based, start and/or end timecode can be provided. In video pictures or images a bounding box of the proglem area can be described.

In some cases, full QC was not performed on an asset. This can be indicated in FullOrPartialQC.

ErrorReference is included to provide a reference to this specific error report.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **QCErrorDescription-type-type** |  |  |  |  |
| ErrorReference |  | Reference tag that can be used to refer to this error instance elsewhere | xs:string | 0..1 |
| ErrorCategory |  | Error Category, in accordance with QC Nomenclature [QCVocab] | xs:string |  |
| ErrorTerm |  | Error Term in accordance with QC Nomenclature [QCVocab] | xs:string |  |
| CategorySpecific |  | Additional data associated with error, based on Error Category. | delivery:QCCategoryError-type | 0..n |
| Comments |  | Any additional comments | xs:string | 0..1 |
| FullOrPartialQC |  | Indicates whether assets was fully evaluated or if evaluation stopped at first error(s) | xs:string | 0..1 |

FullOrPartialQC is encoded as follows [CHS: should this just be a boolean?]

* ‘Full’ – QC was completed
* ‘Partial’ – QC was aborted once error(s) were found. Additional errors may be present.

### QCCategoryError-type

This section contains additional information for errors that are specific to the type of object with an error. Value depends on the QC Nomenclature Category of the error.

Note that definitions are specific to Error Categories (e.g., Video or Audio), and not to specific Error Terms. It is assumed context is sufficient to interpret term-specific data. If not, Best Practices should be developed and/or notes can be put in the Comments field of the parent object.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **DeliveryCategoryError-type** |  |  |  |  |
| Audio |  | Audio Category error specifics | delivery:QCErrorAudio-type | (choice) |
| Video |  | Video Category error specifics | delivery:QCErrorVideo-type |
| TimedText |  | TimedText Category error specifics | delivery:QCTimedText-type |
| Avail |  | Avail Category error specifics | delivery:QCAvail-type |
| Metadata |  | Metadata Category error specifics | delivery:QCErrorMetadata-type |
| Artwork |  | Artwork Category error specifics | delivery:QCErrorArtwork-type |
| Package |  | Package Category error specifics | delivery:QCErrorPackage-type |

[NOTE: Additional Categories are being defined (e.g., “Film”). These will need to be captured here.]

#### QC Utility types

##### QCTimeRange-type

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **QCTimerange-type** |  |  |  |  |
| StartTimecode |  | Track timeline where issue starts.  | manifest:Timecode-type |  |
| EndTimecode |  | Track timeline where issue ends. Omit, if problem persists to end of timeline or if end is unknown | manifest:Timecode-type | 0..1 |

##### QCXMLError-type

Indicates where in an XML document the problem exists. XPath defines the object. Or, if preferred, a line number can reference the object.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **QCXMLError-type** |  |  |  |  |
| XPath |  | XPath reference to object with issue(s) | xs:anyURI | 0..1 |
| LineNumber |  | Line number in file of issue | xs:positiveInteger | 0..1 |

##### QCArea-type

Area of image or picture area where problem exists.

If issue is a single pixel, Width and Height should be 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **QCArea-type** |  |  |  |  |
| XOffset |  | In pixels, x-value of lower left corner of issue.  | xs:decimal |  |
| YOffset |  | In pixels, y-value of lower left corner of issue. | xs:decimal |  |
| Width |  | In pixels, width of picture, inclusive of pixel marked by XOffset. |  |  |
| Height |  | In pixels, height of picture, inclusive of pixel marked by YOffset.  |  |  |

#### QCErrorAudio-type

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **QCErrorAudio-type** |  |  |  |  |
| TimeRange |  | Time range where problem exists. If problem is entire range, do not include this element. | delivery:QCTimeRange-type | 0..1 |
| TimeOffset |  | For errors with alignment issues (e.g., AV Sync), the duration of offset. Negative means audio is ahead of video. | xs:duration | 0..1 |

#### QCErrorVideo-type

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **QCErrorVideo-type** |  |  |  |  |
| TimeRange |  | Time range where problem exists. If problem is entire range, do not include this element. | delivery:QCTimeRange-type | 0..1 |
| Area |  | Area picture where problem exists | delivery:QCArea-type | 0..1 |

#### QCTimedText-type

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
|  |  |  |  |  |
| **QCErrorSubtitle-type** |  |  |  |  |
| TimeRange |  | Time range where problem exists. If problem is entire range, do not include this element. | delivery:QCTimeRange-type | 0..1 |
| TimeOffset |  | For errors with alignment issues (e.g., subtitle Sync), the duration of offset. Negative means subtitle is ahead of video. | xs:duration | 0..1 |
| Text |  | Text that is in error |  | 0..1 |

#### QCErrorAvail-type

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **QCErrorMetadata-type** |  |  |  |  |
| XMLError |  | Reference to location of XML Error | delivery:QCXMLError-type |  |

#### QCErrorMetadata-type

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **QCErrorMetadata-type** |  |  |  |  |
| XMLError |  | Reference to location of XML Error | delivery:QCXMLError-type |  |

#### QCErrorArtwork-type

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **QCErrorArtwork-type** |  |  |  |  |
| Area |  | Area picture where problem exists | delivery:QCArea-type | 0..1 |
| Text |  | Text on image that is in error |  | 0..1 |

#### QCErrorPackage-type

TBD

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **QCErrorPackage-type** |  |  |  |  |
| Subobject |  | Object with package with issue |  | 1..n |

## Logs

A log provides a history of events.

### ProductLog-type

A log is an ordered sequence of events. Log should be ordered from earliest to latest events.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **ProductLog-type** |  |  |  |  |
| Event |  | A reportable event | delivery:ProductLogEvent-type | 1..n |

### ProductLogEvent-type

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Attribute** | **Definition** | **Value** | **Card.** |
| **ProductLogEvent-type** |  |  |  |  |
| EventType |  | Type of event. Can be a Progress Code. | xs:string |  |
| Timestamp |  | Time of event. Should be date or date plus time. | md:YearDateOrTime-type |  |
| Description |  | Description of event | xs:string | 0..1 |
| ErrorReference |  | Reference to a specific error instance as defined in Progress/ErrorDescription | delivery:ProductAvailStatus-type | 0..n |

[CHS: Need to enumerate event types.]

# NOTES

Make sure these are addressed:

* Indication that delivered content isn’t to spec (kind of a waiver).
* Need an indication of what is missing. For example, is forced dubs required for video.
* Ordering something special (e.g., special trailers or artwork)
* Capacity planning and delivery timing?