ONTOLOGY FOR MEDIA CREATION
PART 9: UTILITIES

VERSION 2.6

Motion Picture Laboratories December 16, 2024

Contents

1 Introduction

What is a Utility? “Utility” comes, by way of French, from Latin *utilis*, “useful.” Its range of meaning in modern English is broad, ranging from the general quality of being useful or serviceable through more specific meanings in philosophy, economics, political science, and game theory.[1]

What is the difference between a Utility and Infrastructure? The blurring of the line is most obvious outside the film industry in “utility company” or “utilities” such as water, gas, and electricity suppliers. Those are all, in contemporary usage, utilities, but they are also classed as infrastructure in some circumstances.

In this Ontology, Infrastructure is used in the service of the production – cameras, lights, cloud storage, etc. Utilities are the common bits and pieces used by the Ontology itself, often in multiple different ways. The first recorded use in computing (and there may be earlier ones) is in the IBM Systems Journal in 1962: “The necessary routines such as housekeeping, timekeeping, utility routines, association of equipment, etc.”[2] It is an easy step from applying it to commonly used programs and subroutines to commonly used data structures, which is how it is used here and in the more general technology sector.

1.1 Notational Conventions

In documents generally:

- The definition of a term included in the Dictionary is in bold, followed by the definition, e.g., **Creative Work:** A uniquely identified production.
- When a defined term is used in the text of a document, it is capitalized, for example in “The Production Scene is usually derived from a numbered scene in the Script,” Production Scene and Script are defined in the Ontology. (Note, a word that is part of defined term may sometimes be capitalized by itself as a shorthand, e.g., “Scene” may be used to indicate “Narrative or Production Scene.”)
- References to other Ontology Documents are in ***bold italic***, e.g., ***Part 3: Assets*** or ***Part 3A: Camera Metadata***

For Sample Attributes in the concept documents:

- If a data field or attribute is formally defined in this ontology or a connected ontology, it is italicized, e.g., *Setup* as an attribute refers to a defined concept.
- Attribute […] indicates an attribute can appear more than once, e.g., *Identifier* […]
- →Thing means that an attribute is expressed as a relationship to a Thing, e.g., the →*Script* attribute of Creative Work means there is a relationship Creative Work→*Script*

[1] As well as the archaic term “utility actor,” “an actor of the smallest speaking parts in a play.” For this and other compounds, see the Oxford English Dictionary, s.v. “utility.”
[2] OED, s.v. “utility”

- A combination of the two indicates that the concept can have relationships to a set of things, e.g., →Components […]
- Many elements of the Ontology have a Context element. (See **Part 2: Context**.) Relationships declared in the Context are implied to have the item to which the Context is attached as their starting point, for example, Narrative Location→Context→Narrative Scene.

  Contextual relationships that are especially important to the concept being defined are given in the sample attributes tables as C→Thing or C→Thing […] as appropriate. These relationships can just as well be on the object that has the Context. For example, if Narrative Location has "C→Narrative Scene" as an attribute, it is ok to have the relationship directly on the Narrative Location or in its Context, e.g. Narrative Location→Narrative Scene or Narrative Location→Context→Narrative Scene.

  Some implementations (e.g. RDF) place these relationships directly on the class as well as allowing them in Context, and others (e.g. JSON) place all relationship in a Context.

2 Utilities

Utility: Common data model or data structure used in multiple places and in multiple ways in a larger system.

This section will grow as the Ontology grows. Obvious candidates for inclusion are structures and formats for times and dates, for example. The intent is to use existing standards and ontologies wherever possible and include them by reference.[3]

2.1 Identifier

Identifier: A string of characters that uniquely identifies an object within a particular scope.

Identifier Scope: The universe within which an identifier is valid and unique.

At its simplest, an Identifier is just a way of referring to something – a kind of name for it. Examples include an ISBN for a book, and social security number for a person subject to the US tax code, and an EIDR ID for a TV series. The thing an Identifier refers to is called its referent, and the act of converting an Identifier to its referent[4] is called "resolution" or "resolving." To be useful, an identifier should have only one referent. This is true for the examples given below.[5] More precisely, an Identifier should have a single referent within its Identifier Scope.

Some identifiers (e.g., DOI and EIDR) carry the Identifier Scope as part of the Identifier, but others need to declare the Identifier Scope explicitly. For example, both South Carolina and New Mexico driver's license numbers consist of 9 decimal digits, and they are only unique within their scope – the issuing

[3] For example, ISO, W3C, and IETF provide many of the things that are needed for times, dates, countries, and languages.

[4] Or information about its referent

[5] With exceptions for occasional errors, of course.

state. As another example, "42" is a perfectly good identifier. If the scope is the AlloCine film database it is *L'Affaire est dans le sac*, at the AlloCine series database it is *Birds of Prey*, and at Česko-Slovenská filmová database it is *Love and Death*.[6]

It is helpful, but not required, for a referent to have only one Identifier within an Identifier Scope.

Sample Attributes for Identifier

Term	**Description**
Identifier Value	The identifier itself.
Identifier Scope	See above.
Combined Form	Optional. See Notes.
Resolved Identifier	Optional: A URI that resolves to the Identifier's referent

Notes:

Some systems impose restrictions on Identifier Value. From most restrictive to least restrictive, these can be alphanumeric only, ASCII-only, Subset of Unicode (e.g. excluding emojis), all of Unicode, and arbitrary binary – though the last can raise interoperability issues when serializing it.

It is often convenient to combine the Identifier and Identifier Scope in a single string. There are several ways of doing this, but the most common use URN (URN:ID:IdentifierScope:Identifier) or URI path components (Identifier Scope/Identifier). The choice of how to combine them is often dictated by a resolution service.

Resolved Identifier is a URI that resolves to the referent for the Identifier/Scope. It can be included in OMC data that is sent to applications that do not know how to resolve identifiers and so require a URI rather than an Identifier/Scope. It can also be used more persistently to cache the URI for a resolved identifier; this has all the advantages (performance, simplicity for a common case) and disadvantages (knowing when to regenerate the value when the resolution changes) of caches in general.

RFID tags and barcodes are examples of Identifies for Physical Assets.

It is perfectly acceptable for an item to have more than one identifier; these multiple identifiers usually are in different Identifier Scopes[7]. Different systems can create their own identifiers for internal reasons. For example, a struct RDF system may want to create its own identifiers when it imports data from other sources; when it makes the information available to other systems, it should include its own identifier in the list of identifiers for an object. Similarly, an animation tool

[6] https://www.allocine.fr/film/fichefilm_gen_cfilm=42.html , https://www.allocine.fr/series/ficheserie_gen_cserie=42.html , and https://www.csfd.cz/film/42

[7] There may be more than one identifier in a scope if, for example, the system or systems responsible for that scope have an aliasing mechanism to cope with duplicate entries.

that has its own internal asset management system can add its own identifiers to externally generated objects. People often have multiple identifiers – one in an HR system, one in a guild system, one in a production staff database, and so on and keeping all of these with the Person record in OMC makes it easier for other applications to find the person.

2.2 File Details

Not all systems use identifiers. Many applications and formats[8] refer to things entirely through filenames, which include a file path. A filename can be used instead of an identifier within the scope of a particular application and the environment within which it is running: it uniquely determines the identity and location of the resource to which it refers. Outside of that application and environment, the filename may be ambiguous or non-resolvable, which complicates sharing with other applications. In OMC implementations, it is not uncommon to have a filename for an Asset Structural Class and an Identifier, managed by a MAM, for the Asset to which it is connected.

Note: OMC uses the word “file” for data outside the Ontology itself, whether that is essence data or blobs of non-OMC metadata. Historically, this has been stored in a file system as noted above. For simplicity OMC uses the word “file” to encompass other storage as well, such as cloud storage and networked storage.

Everything in OMC requires an Identifier. File Details should be used in addition to identifiers, not as a replacement for them.

Sample Attributes for File Details

Term	**Description**
File Name	The name of the file, as understood by the storage system
File Path	A path for the file. Besides being //<…> for a typical file path, it can also be file:… or https:…
File Extension	Optional. The extension part of the File Name. This is for convenience if there is a full file path, since the extension is also included in the file name; in general, it can also be used to decide what application(s) can use the Asset or by an application to determine the correct way of loading the Asset, even for systems that are not file-based.
Media Type	Optional. This describes the type of Asset Structural Class, for example with a MIME type.

Notes:

[8] Such as ProTools for audio, or MaterialX for CG material files. USD is file based, but supports plugins that can handle identifiers.

If there is no MIME type of appropriate granularity, applications in the workflow should come to agreement on a list of other values. In most cases, though, information in an associated Asset Functional Class should provide the necessary information.[9]

Many workflows will need a File Type when to decide which application(s) can be used with the Asset, even if they are not file-based.

2.3 Location

Location: A particular place or position either in either the real world or the narrative world.

Sample Attributes for Location

Attribute	**Description**
Identifier […]	One or more Identifiers for the Location. This Identifier is for the Location itself and may be different from the Identifier for the thing that uses this location (such as a Narrative Location or a Production Location.[10])
Name	The name of the Location, independent of the use of the Location.
Description	A description of the Location, independent of the use of the Location.
→ Related Location […]	Locations to which this location is related. This field is primarily intended for connecting less detailed locations, e.g., a building, to more detailed locations, e.g., a sound stage.
Address	The address of the Location. There are many formats available for this, and a future version of the Ontology will provide more detail on the contents of this field and how to use it with existing standards.
Coordinates	Geolocation coordinates for the Location. There are many formats available for this, and a future version of the Ontology will provide more detail on the contents of this field and how to use it with existing standards.
Custom Data	Anything that is application or workflow dependent that can't be otherwise expressed in the Ontology or needs to be present in a particular format.

Notes:

This Location is intentionally generic. Its principal use is as a component of other elements of the Ontology, such as Narrative Location and Production Location.

These Locations may exist in a system that is managed and run independently of the production process. The Identifier field refers to the Location within the scope of those systems. Ontology elements that use Location will usually have their own Identifiers but could use an Identifier of

[9] For instance, there is no need to extend application/pdf to cover a Script, since that is determined by the Asset's Functional Class.

[10] See ***Part 2: Context***

the Utility Location if they really need no other information than is contained in those external systems. For example, the identifier Q73094 with Identifier Scope “wikidata.org” can be used in the Location for The Queen’s College, Oxford, which was used as a Production Location in *The Golden Compass* (2007.) That Production Location has its own Identifier(s) (in a studio or production database, for example) each with its own Identifier Scope.

2.4 Compositions

Many Assets are generated from a combination of Assets and instructions, using specialized tools. For example, an Editorial Sequence is generated from multiple pieces of video and a timeline with instructions (such as an EDL.) The Editorial Sequence itself is not a Video Asset, but the tool that manipulates it can produce a new Asset (video) as output. The same is true for a STEM, which is typically a combination of multiple audio tracks and an audio session, the product of which is itself a new playable piece of audio. A full CG model is composed of geometry, materials, maps, and other components; that model can be rendered as a turntable video, a preview image, or used as part of a final animation.

In OMC, these are all Compositions. A Composition is not itself an Asset, but can produce one or more kinds of Asset. Compositions can include other Compositions. An Editorial Sequence can include subsidiary sequences (which are themselves compositions), and a CG model can be composed of, for example, a character and the prop she is holding.

Subclasses of composition can use specialized naming for their data elements, e.g. “Shot” and “SCD” rather than undifferentiated Assets for an Editorial Sequence.

Composition: A set of Assets or other Compositions that can be combined to produce a new Asset

Compositions can be turned into Assets by applications as needed by the workflow, as when an Editorial Sequence is used to produce a digital video Asset for review.

Compositions included in another Composition must be of an appropriate type. It makes no sense to have a CG Material in an editorial sequence. This is hard to enforce with current schema technology, so applications that generate or consume Compositions should check accordingly.

Not all Compositions produce an output. This is especially true for assemblies of computer graphics material. For example, a Material may only ever be used as part of some larger composition, and an individual component of a Geometry Assembly (see ***Part 3D: CG Assets***) might only ever be rendered when included in that Geometry Assembly.[11]

Compositions can be rendered in multiple ways - they describe the assembly of pieces, not the output of that assembly. For example, a CG Composition can be rendered as a single image, as a turntable or as differing qualities of rendering or output video. A single audio STEM can produce a stereo mix or a 5.1

[11] Although many such assets will have at least one rendering in their lifetime, for the review and approval process, the rendering may be ephemeral, or not done at all.

mix. It is the application[12] assembling the Composition that determines the output, not the Composition itself.

One of the Assets in a Composition should be a set of instructions for combining the other pieces, much like the instruction sheet in flatpack furniture or a Lego model. For some Compositions, such as audio and video compositions, this is a special file type. For other Compositions, such as a model defined with a set of USD files, the instructions are implied by the first file opened.

Sample attributes for Composition

Term	**Definition**
Identifier […]	One or more identifiers for the Composition. At least one of these should be resolvable within the production environment.
Name	The name of the Composition.
Description	A description of the Composition.
Asset ->[] *Asset Structural Class*->[]	The Assets that make up the Composition. For many compositions these can also be Asset Structural Classes.
Composition ->[]	Any included compositions, such as a Material for a CG Model.
->startHere	Starting point for assembling the Composition. This is often the first thing that an application loads. See above for examples. This is an Asset.
Custom Data	Anything that is application or workflow dependent that can't be otherwise expressed in the Ontology or needs to be present in a particular format.
→ *Context* […]	Any Context for the Asset. See ***Part 2: Context***

Notes:

A Composition's relationship to any Assets it produces is hasProduct/productOf

[12] Often represented by an OMC Task

A Composition may not have a product, for example if it is always included in a larger Composition.

A Composition's relationship to its starting point is hasStart/startFor

The relationship for all inclusions is includes/includedBy. Subclasses of Composition may have separate fields for different types of inclusions.

Inclusions should only be items used directly by the Composition, not items included in included Compositions.

In practice, it is more common to use subclasses of Composition for particular parts of the workflow rather than this root class. These subclasses generally call out items of particular interest for a workflow, such as Geometry and Materials for a CG Assembly (See ***Part 3D: CG Assets***.) Implementations with full subclassing features, such as RDF, will usually specify those as new fields and subclassed relationships. Implementations without subclassing or where emulation of subclassing is difficult, such as JSON, can put all of those into the generic fields in the base Composition, as long as any data provided in that format includes enough information about the inclusions for a recipient to deduce how it is intended to be used.

The list of Assets included in a Composition can act as a manifest when handing a CG Assembly over to another part of the workflow.

3 Common Data Types

Simple data types should use existing standards wherever possible. Some of the more common ones are:

3.1 Countries and Languages

- Countries: Use ISO 3166
- Languages: Use IETF BCP 47, with the practices published as the Language Metadata Table (LMT) at https://www.mesaonline.org/language-metadata-table

3.2 Dates, Times, and Durations

- Date, Time, and DateTime: Use ISO 8601
- Duration: Use ISO 8601
- Time intervals: Use ISO 8601, using Start/End or Start/Duration whenever possible. Duration only is acceptable but requires extra context to be useful. Duration/End is allowed but probably less common.

Future versions of this document will include recommendations for timestamps and timecodes, based on industry feedback.

3.3 Measurements

Measurements are important throughout the production process. Characters have approximate heights and weights, images have a size in pixels, inches, or millimeters, production sets and props have real dimensions, and the coordinate system units in computer graphics often have to be translated into real-world measurements when the object is used.

All measurements have two pieces: the units of the measurement, and the number of those units. This can be expressed in two ways; not all implementations will support both.

3.3.1 Units

Units should be expressed as follows:

- Metric system: use SI standard abbreviations, such as “kg” and “g” for weight and “mm” and “m” for length.
- Pixels: measurements expressed in pixels should use “pixels” in the units field.
- Imperial measure: These should use the abbreviated form of the unit, e.g. “mi” not “mile” and “in” not “inches”. Hundredweight and ton should be avoided because of the confusion between their long and short versions.[13]

3.3.2 Measurement as typed data

Attribute	Description
quantity	The numeric part of the measurement, specific as a number, not a string
units	The units of the measurement.

Notes:

Measurements expressed in feet and inches should be converted to inches, and those in pounds and ounces should be converted to ounces.

3.3.3 Measurement as a formatted string

The form for metric measurements is “<n>km<n>m<n>cm<n>mm” for length, and similarly for weight.

The form for pixels is “<n>px”.

The form for imperial measurements is “<n>mi<n>ft<n>in”.

3.3.4 Measurements without units

Physical measurements should always include units. For some digital assets, units may not be necessary – for example, Digital Images are almost always measured in pixels. For consistency and interoperability,

[13] “Let’s just say we’d like to avoid any imperial entanglements” – Obi-Wan Kenobi in *Star Wars: Episode IV – A New Hope* (1977)

it is worth adding units even in these cases, although some systems that generate OMC may have issues with adding them in the short term.

3.4 Dimensions

Rectangles and rectangular cuboids are extremely common, and OMC provides a utility class for them.

Dimensions: The linear extents of a rectangle or rectangular cuboid

Attributes for Dimensions

Attribute	**Description**
width	The width of the object. This is often thought of as being along its X-axis.
height	The height of the object. This is often thought of as being along its Y-axis.
depth	Optional. The depth of the object, often thought of as being along its Z-axis. See Notes.

Notes:

> Most Images are two-dimensional and for those, depth should not be used for things like bit depth or bits per pixel. Rather, depth indicates that there really is another “axis” of pixels, thinking of each layer or plane in that direction as another “slice” of a three-dimensional set of pixels. This is used to support volume textures, for example.

3.5 Annotations and Tags

In spite of the best efforts of ontologists, free form data is necessary and useful in the real world. OMC supports this with some utility classes.

In OMC, anything that exists as an identified entity – such as Assets, Participants, Locations, and Infrastructure - can have one or more annotations attached. For example, a piece of Concept Art can have annotations from the Production Designer and the Director, indicating reasons for liking or disliking it. The same piece of Concept Art can have multiple sets of tags, for example one generated by AI with attributes it deemed relevant, such as mood, darkness, notion of genre, and so on, and another set provided by the art department as part of their database search methods.

In the JSON implementation, annotations and tags apply to Entities (as it is defined in the schema) and its subtypes. In RDF they apply to AnythingWithAnIdentifier and its subclasses.

3.5.1 Annotation

Sometimes adding a short explanation or extra information to something is necessary or useful. In the production process this is formalized in some situations, such as a script supervisor’s notes. Those notes have an existence on their own and are OMC Assets. However, it is often usefulto add commentary to a particular item. It is also the case that multiple people can add notes to things – the production process is collaborative. This extra information is an OMC Annotation.

Annotation: Human readable commentary, explanation, or information.

Attributes for Annotation

Attribute	Description
title	String: A title for the note
text	String: the text of the note
->author	Participant or, if unavoidable, a string.

Notes:

Different titles can help distinguish particular Annotations.

Annotations are intended for human readable content and should not contain serialized data; that should be kept in Custom Data fields.

3.5.2 Tag

A tag is a term assigned to a piece of information, in OMC an object is anything that has an Identifier. Individual tags are generally short, such as “noir”, “explosion”, or “manic.” Tags come from a set of (usually) pre-defined strings from a particular tagging system, such as a taxonomy, list of controlled vocabulary, or a folksonomy, and help with organizing and searching data. For example, a set of genre tags can come from IMDb or Wikidata, and machine-generated tags come from a particular piece of software. OMC call these sets domains.

Tag: A short string from a particular set, used for categorization and description

Attributes for Tag

Attribute	Description
domain	String: An indication of the set or system in which the tag values are relevant or defined
->value[]	String or Number: a set of tags taken from the domain.

Notes:

Tags can be overused, and every system and organization has different rules and terminology governing them. This can limit their usefulness in workflows that extend across different systems and organizations, indeed OMC in no small part exists because of the problems created by tags. However, it might still be useful to include them when exchanging data, for example AI systems are likely to be better able to handle and reconcile tags over more traditional systems.

3.6 Mathematical Types

Modern filmmaking needs a lot of math. OMC defines some of the more useful types, and more will be added based on implementation needs.

3.6.1 Matrices

Matrices are used extensively in computer graphics and in some kinds of color management.

OMC does not currently define a general Matrix class, but does have the special cases of a 3x3 matrix and a 4x4 matrix for CG work. A general matrix can be implemented as a use of Structured Data or Structured Document.

Matrices are in row-major order and applications that use column-major order can convert them on input and output.

Matrix33: A square matrix with three rows and columns.

Matrix44: A square matrix with four rows and columns.

Notes:

> Representing the values of the matrix is dependent on the serialization language. JSON and RDF have different conventions, and rather than forcing a general serialization, OMC defers to whatever is natural for the implementation

3.6.2 Points

These definitions are not intended to replicate the contents of various point-oriented CG Assets (see ***Part 3D: CG Assets***), but can be used for describing relatively more constrained

Point2: A point with two coordinates.

Point3: A point with three coordinates.

3.7 Useful Terms and Concepts

OMC occasionally uses terms on their own, without providing any data structures. In these cases, it is clarity of concept that matters rather than any particular implementation. These terms are collected here.

Codec: A system that compresses or decompresses digitized audio, video, or images.

Notes:

> Different parts of the production process use different Codecs, and lists of codec types are in the relevant part of OMC.
>
> Analog to digital conversion is often done by a separate system, and is not included in OMC.

isSelfContained: An Asset that does not depend on any other assets for a particular functional use.

Notes:

> This should be a Boolean value. Having the property with no value is allowed but discouraged, as is using empty strings.
>
> This property can exist on any Asset functional class. It is explicitly included for a Material and CG Model, but may also be useful for an Asset whose structural class is known to be self-contained, such as a USDZ file.
>
> Some assets might be self contained in one situation and not in another; in this case a new Asset Functional Class can be created and attached to the same Asset Structural information.

Appendix A External Definitions

These are terms defined elsewhere in the Production Ontology, included here for ease of reference.

Media Creation Context: Informs scope within the construction process of a Creative Work.

> See **Part 2: Context**

Asset: A physical or digital object or collection of objects specific to the creation of the Creative Work.

> See **Part 3: Assets**

Camera Metadata: Capture-specific details and information about the Camera itself.

> See **Part 3A: Camera Metadata**

Participant: The entities (people, organizations, or services) that are responsible for the production of the Creative Work.

See **Part 4: Participants**

Task: A piece of work to be done and completed as a step in the production process.

See **Part 5: Tasks**

Creative Work: A uniquely identified production.

See **Part 6: Creative Works**

Relationship: Describes and defines the connections between elements of the Ontology, such as Assets, Tasks, Participants, and Contexts.

See **Part 7: Relationships**

Infrastructure: The underlying systems and framework required for the production of the Creative Work; it is generally not specific to a particular Creative Work.

See **Part 8: Infrastructure**

Utilities: Common data models and data structures used in multiple places and in multiple ways in a larger system.

See **Part 9: Utilities**

Identifier: An identifier uniquely identifies an entity within a particular scope.

See **Part 9: Utilities**