UC3Merritt

University of California Curation Center

Merritt Object Modeling

Rev. 0.12 – 2010-11-03

1 Introduction

Information technology and resources have become integral and indispensable to the pedagogic mission of the University of California. Members of the UC community routinely produce and utilize a wide variety of digital assets in the course of teaching, learning, and research. These assets represent the intellectual capital of the University; they have inherent enduring value and need to be managed carefully to ensure that they will remain available for use by future scholars. Within the UC system the UC Curation Center (UC3) of the California Digital Library (CDL) has a broad mandate to ensure the long-term usability of the digital assets of the University. UC3 views its mission in terms of *digital curation*, the set of policies and practices aimed at maintaining and adding value to authentic digital assets for use by scholars now and into the indefinite future [Abbott].

In order to meet these obligations UC3 is developing Merritt, an emergent approach to digital curation infrastructure [Merritt]. Merritt devolves infrastructure function into a growing set of granular, orthogonal, but interoperable micro-services embodying curation values and strategies [Foundations]. Since each of the services is small and self-contained, they are collectively easier to develop, deploy, maintain and enhance [Denning]; equally as important, since the level of investment in and commitment to any given service is small, they are more easily replaced when they have outlived their usefulness. Yet at the same time, complex curation functionality can emerge from the strategic combination of individual, atomistic services [Fisher].

NOTE The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” are to be interpreted as described in RFC 2119 [RFC2119].

2 Object modeling

Digital content curated in a Merritt environment typically will be managed by the Merritt Storage service [Storage]. The Storage service imposes minimal structural requirements on the digital content that is managed in it; in particular, Storage-managed content is not typed so the service has no understanding of object semantics. Additional requirements are necessary to facilitate the management of object semantics. These requirements will be enforced by the Ingest service [Ingest] and the resulting object semantics will be exposed via the Inventory service [Inventory].

2.1 What is an object?

Merritt defines a *digital object* (often simply referred to as an *object*) as the representation in digital form of a *thing*. The term “thing” is used advisedly in order to be essentially unlimited in scope. For

example, Merritt objects can correspond to a variety of important classes of things, some more tangible than others, including:

- Concrete physical objects, such as books, slide images, museum artifacts, films, newspapers, traditional scholarly publications, field or laboratory notebooks, etc.
- Born-digital objects, such as online publications, websites, electronic theses and dissertations, spreadsheets, databases, spreadsheets, etc.
- Aggregate objects, such as curatorially- or administratively-defined sets of objects.
- *Seed* objects, which have a preservation–ready identifier and perhaps some minimal metadata, but for which Merritt currently has no associated content. Seed objects meet the needs of curators to obtain a "placeholder" for in-process content, from the moment of conception to the moment of "birth" (e.g., initial publication or first deposit in Merritt), a process that may span from weeks to years in various workflows.
- Standardized vocabulary objects, perhaps with associated URIs for semantic web compliance, such as canonical format names, Namaste directory types, Checkm schema profiles, licenses, policy statements, etc.

Many other classes of objects are possible as well. The Merritt object model is designed to encompass the widest possible range of content.

Content curators have full control over the granularity of the mapping from a unit of curatorial content to one or more objects. For example, a series of 12 annual conference proceedings can be deposited as a single "series" object, as 12 individual "conference" objects, or as 385 individual "paper" objects. UC3 can advise curators on the tradeoffs and methods for defining their assets into objects for submission. Every Merritt object can thus be seen loosely to represent a coherent intellectual *work*, c.f. [FRBR], with coherence being a function of a particular curatorial context.

2.2 Classes and collections

Merritt objects MAY be aggregated together into *classes* (or more colloquially, collections) to meet various curatorial and administrative needs. Classes themselves MAY be members of other classes to form acyclic hierarchical inheritance structures. Class membership can be defined in one of two ways:

- An object can assert its membership in the class.
- A class can assert its object members.

Any given class MUST use one or the other assertion mechanisms for all of its members.

Within each class hierarchy of which it is a member, an object SHOULD explicitly assert its membership only in the innermost enclosing parent class; class membership in all higher-level parent classes of the hierarchy is asserted implicitly through structural inheritance.

Classes are represented in Merritt by special *class objects*. Class membership is thus asserted by references from objects to the class objects of which they are members, or by references from class objects to all of the objects that are their members.

NOTE The term "collection" is widely used in many curation and preservation contexts with similar, but not always equivalent meanings. Therefore the more generic term "class" is used in its stead.

2.3 Object properties

Each Merritt object is associated with the following REQUIRED properties:

- A unique *primary identifier* in the ARK scheme [ARK]. This identifier MAY be supplied by the submitting curatorial agent; if not, it SHALL be assigned automatically by the Ingest service [Ingest].
- An object *type*, indicating its nature with the Merritt environment (see Table 1).
    - *Curatorial* objects are those representing content originating from external contributors.
    - *System* objects are those necessary for the internal operation of the Merritt curatorial environment.

Type	***Definition***
`MRT-curatorial`	A Merritt *curatorial* object. Curatorial objects are those representing content originating from external curatorial contributors.
`MRT-system`	A Merritt *system* object. System objects are those necessary for the internal operation of the Merritt curatorial environment.

Table 1 – Merritt object types

NOTE All Merritt property names and values that start with "Merritt" or "MRT", on a case-insensitive basis, are reserved.

Additional object types MAY be defined as necessary.

- An object *role*, indicating its particular function within the context of its type (see Table 2).
    - *Class* objects represent aggregations of objects. Class objects MAY be of type "`MRT-curatorial`" or "`MRT-system`";
    - *Content* objects represent individual abstract works. Content objects MUST be of type "`MRT-curatorial`".

Role	Definition
`MRT-class`	A Merritt *class* object. Class objects represent arbitrary, but nevertheless administratively- or curatorially meaningful sets of objects.
`MRT-content`	A Merritt *content* object. Content objects represent individual abstract works.

Table 2 – Merritt object roles

Additional object roles MAY be defined as necessary.

- *Owner* class membership. Ownership represents a domain of ultimate responsibility for content managed in a Merritt environment, including acceptance of all financial obligations arising from that management. This property is defined by a reference *from* the object *to* the owner class of which it is a member.
- Curatorially-defined *collection* class memberships. This property is defined by a set of references *from* the object *to* the collection classes of which it is a member.
- Descriptive metadata in the form of the Dublin Kernel REQUIRED "who", "what", "when", and "where", and OPTIONAL "how" and "why" elements [Kernel]. The "who" element roughly corresponds to the object's creator; "what" to its title; "when" to its creation date; "where" to one or more format identifiers; "how" to a methodological statement; and "why" to an expository note. This description is intended to provide a useful, but relatively simple overview of the object. In particular, it is *not* intended as a substitute for complete cataloging, which is expected to exist external to the Merritt curation environment, preferably at a location specified by a local identifier.

In addition, class objects (of role "`MRT-class`") MUST also be associated with the following property:

- An object *aggregate*, indicating the aggregate basis for the class (see Table 3).
    - Curatorially-defined *collection* classes represent a common intellectual context for its members.
    - *Owner* classes represent domains of ownership responsibility.
    - *Service level agreement* classes represent service-level agreements.

Aggregate	Definition
`MRT-collection`	A Merritt *collection* class. Collections define curatorially meaningful sets of objects.
`MRT-owner`	A Merritt *owner* class. Ownership classes represent domains of ultimate responsibility for content managed in a Merritt environment.
`MRT-service-level-agreement`	A Merritt *service level agreement* class. SLA classes represent domains of coverage for agreements between customers and service providers.

Table 3 – Merritt class object aggregates

Additional class object aggregates MAY be defined as necessary.

Each owner class MUST be the child of some parent service level agreement class. Collection classes MAY be the children of other parent collection classes. The collection and service level/owner hierarchies MUST be kept orthogonal and independent of one another.

In addition, class objects (of role "`MRT-class`") MAY be associated with the following property:

- Class *members*. This property is defined by a set of references *from* the class *to* the individual objects that are its members.

In addition, any Merritt object MAY be associated with the following OPTIONAL properties:

- One or more local identifiers. These identifiers SHOULD be meaningful in the local context of the object's curator. In general, they SHOULD resolve to a more detailed descriptive catalog record for the object external to the Merritt curation environment.

Additional object properties MAY be defined as necessary.

2.4 Object components

A Merritt object is composed of an arbitrary number of *versions*, each representing a discrete state of the object and further composed of an arbitrary number of *components* or *files*, c.f. [Dflat]. Merritt defines four special files expressing significant object properties that MUST be components of every Merritt object:

- MOM (Merritt object model) metadata [`mrt-mom.txt`]. The MOM file component is used to define the fundamental properties of the object, including primary identifier, type, role, aggregate (if a class object), and (optional) local identifiers, expressed in ANVL form:

```
  primaryIdentifier: identifier
  type: MRT-curatorial | MRT-system
  role: MRT-class | MRT-content
[ aggregate: MRT-collection | MRT-owner |
             MRT-service-level-agreement ]
[ localIdentifier: identifier
  ... ]
```

  NOTE Brackets `[` and `]` enclose optional elements; a vertical bar `|` separates alternative choices; and an ellipsis `...` indicates 0 or more repetitions of the previous element.

- Ownership class [`mrt-owner.txt`]. The owner file component is used to define the object's owner, that is, the individual or corporate agent accepting final responsibility for the object, including all financial obligations arising from the object's management in Merritt. The owner is represented in the "`mrt-owner.txt`" file by the primary ARK identifier of the

owner class object, which MUST be of type "MRT-system", role "MRT-class", and kind "MRT-owner":

```
owner
```

- Membership classes [mrt-membership.txt]. The membership file component is used to define the set of one or more curatorially-defined collections of which the object is a member. These collections are represented in the "mrt-membership.txt" file by the primary ARK identifiers of the collection class objects, which MUST be of role "MRT-class" and kind "MRT-collection":

```
collection
...
```

- ORE resource map [mrt-object-map.ttl]. The resource map file component is used to define the object as an *aggregate* of *resources* (i.e., files) in terms of RDF triples [ORE, RDF]. For file components that are *metadata*, as opposed to primary content, the resource map also defines the metadata schema and serialization syntax. The resource map is expressed in Turtle form [Turtle]:

```
@prefix mrt: <http://uc3.cdlib.org/ontology/mom#>.
@prefix ore: <http://www.openarchives.org/ore/terms/>.
<object-url>                          # This object is ...
  ore:aggregates                      # ... an aggregate of ...
    <mom-url>,                        # MOM metadata
    <owner-url>,                      # Owner class
    <membership-url>,                 # Collection classes
    <resource-map-url>,               # Resource map
  [ <erc-url>, ]                      # (Optional) ERC metadata
    <file-url>,                       # Other object files
    ...;
  mrt:hasMetadata                     # ... has ingest metadata
    <ingest-url>;
  mrt:hasMetadata                     # ,,, has membership metadata
    <membership-url>;
  mrt:hasMetadata                     # ... has MOM metadata
    <mom-url>;
  mrt:hasMetadata                     # ... has owner metadata
    <owner-url>;
[ mrt:hasMetadata                     # ... has ERC metadata
    <erc-url> ] .
<mom-url>                             # MOM schema and MIME type
  mrt:metadataSchema
    "MRT-MOM";
  mrt:mimeType
    "text/anvl".
```

```
  <owner-url>                          # Owner schema and MIME type
    mrt:metadataSchema
      "MRT-owner";
    mrt:mimeType
      "text/plain".
  <membership-url>                     # Collections schema and MIME
    mrt:metadataSchema
      "MRT-membership";
    mrt:mimeType
      "text/plain".
  <resource-map-url>                   # Resource map schema, MIME
    mrt:metadataSchema                 # type, and description
      "MRT-ORE";
    mrt:mimeType
      "text/turtle";
    ore:describes
      <object-url>.
[ <erc-url>                            # ERC schema and MIME type
    mrt:metadataSchema
      "ERC";
    mrt:mimeType
      "text/anvl". ]
```

NOTE Resource map examples use the Turtle "`,`" and "`;`" abbreviation symbols. It is possible, and acceptable, to express functionally equivalent resource maps using more verbose alternative expressions.

NOTE The ORE specification currently only defines serializations in RDF/XML, RDFa, and Atom/XML. Thus, Turtle is a non-standard ORE serialization. However, Turtle has the significant benefit of far greater transparency for the human reader and its use is expected to be sanctioned by the ORE specification in the future.

NOTE Until such time as a formal MIME type for the ANVL format is established at the IANA registry, the experimental MIME type "`text/x-anvl`" SHOULD be used.

- ERC metadata [`mrt-erc.txt`]. The ERC file component is used to define a minimal set of descriptive properties of the object, based on the Dublin Kernel "who" (creator), "what" (title), "when" (date), "where" (identifier), and "why "(note) elements.

```
  erc:
  who: creator
  what: title
  when: date
  where: identifier
[ where: identifier
  ... ]
[ how: methodology ]
[ why: note ]
```

The REQUIRED "where" element appearing in the ERC file SHOULD be the primary ARK identifier for the object; any subsequent OPTIONAL "where" elements SHOULD be local identifiers.

Required element values that cannot be supplied MUST use one of the pre-defined coded values appropriate to the nature of the omission, for example, "`(:unas)`" (unassigned), "`(:unkn)`" (unknown), or "`(:null)`" (meaningfully empty), c.f. [ERC].

Additionally, class objects (of role "MRT-class") MAY have the following component:

- Class *members* [`mrt-members.txt`]. The members file component is used to define the objects that are members of the class. These objects are represented in the "mrt-members.txt" file by their primary ARK identifiers:

  object
  …

Additional properties MAY be associated with the object by adding additional metadata components. The semantics of these components SHOULD be declared using one of the schema identifiers in Table 4 with a "`mrt:hasMetadata`" predicate in the object resource map. Similarly, metadata syntax SHOULD be declared using the appropriate MIME type with a "`mrt:mimeType`" predicate. Object producers with latitude regarding metadata file names SHOULD use the reserved file names associated with the various schemas.

Identifier	***File name*** *(recommended)*	***Metadata schema***
AES-X098B	mrt-aes-x098b.xml	AES-X098B audio technical metadata.
CreativeCommons	mrt-creative-commons.xml	Creative Commons rights metadata.
DublinCore	mrt-dublin-core.xml	Dublin Core metadata.
ERC	mrt-erc.txt	Electronic Resource Citation descriptive metadata.
JHOVE2	mrt-jhove2.xml	JHOVE2 characterization metadata,
MARCXML	mrt-marcxml.xml	MARC bibliographic metadata.
MIX	mrt-mix.xml	MIX / NISO Z39.87 still image technical metadata.
MODS	mrt-mods.xml	MODS descriptive metadata.
MRT-content-model	mrt-content-model.txt	Merritt content model metadata.
MRT-ingest	mrt-ingest.txt	Merritt ingest metadata.
MRT-members	mrt-members.txt	Merritt class member assertions.
MRT-membership	mrt-membership.txt	Merritt class membership assertions.
MRT-MOM	mrt-mom.txt	Merritt object model metadata.
MRT-ORE	mrt-object-map.ttl	Merritt ORE resource map.
MRT-owner	mrt-owner.txt	Merritt owner metadata.
MRT-service-level-agreement	mrt-service-level-agreement.txt	Merritt service level agreement metadata
PREMIS-object	mrt-premis-object.xml	PREMIS object preservation metadata.
PREMIS-rights	mrt-premis-rights.xml	PREMIS rights preservation metadata.
VRA-core	mrt-vra-code.xml	VRA Core cultural heritage metadata.

Table 4 – Metadata schemas

NOTE Within a Merritt curation environment all file system names beginning with "merritt" or "mrt" (on a case-insensitive basis) are reserved.

Additional metadata schemas MAY be defined as necessary.

3 Ontological classes and relationships

Merritt object resource maps make use of ontological classes and properties defined in the following namespaces (see Table 5).

Prefix	*URI*	*Ontology*
cc	http://creativecommons.org/ns#	Creative Commons rights metadata.
dc	http://purl.org/dc/elements/1.1/	Dublin Core metadata elements.
dcterms	http://purl.org/dc/terms/	Dublin Core metadata terms.
ore	http://www.openarchives.org/ore/terms	ORE aggregation terms.
mrt	http://uc3.cdlib.org/ontology/mom#	Merritt classes and properties.
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	RDF vocabulary terms.
rdfs	http://www.w3.org/2001/01/rdf-schema#	RDF Schema vocabulary terms.
xsd	http://www.w3.org/2001/XMLSchema	XML Schema datatypes.

Table 5 – Merritt namespaces

Additional namespaces MAY be defined as necessary.

Merritt defines a number of its own ontological classes and properties, all in the "mrt" namespace (http://uc3.cdlib.org/ontology/mom#).

3.1 Ontological classes

The following Merritt ontological classes are defined.

mrt:Component	
rdfs:label	Component
rdfs:comment	Abstract Merritt class for object model components: object, version, file.
rdf:type	rdf:Class

mrt:File	
rdfs:label	File
rdfs:comment	Merritt class for files. A file is a formatted octet stream.
rdf:type	rdf:Class
rdfs:subClassOf	mrt:Component

mrt:Object	
rdfs:label	Object
rdfs:comment	Merritt class for objects. An object is the representation in digital form of an abstract work. An object is composed of an arbitrary number of versions.
rdf:type	rdf:Class
rdfs:subClassOf	mrt:Component
rdfs:seeAlso	mrt:Version

mrt:Version	
rdfs:label	Version
rdfs:comment	Merritt class for versions. A version is a set of files that collectively represents a discrete state of an object. A version is composed of an arbitrary number of files.
rdf:type	rdf:Class
rdfs:subClassOf	mrt:Component
rdfs:seeAlso	mrt:Object
rdfs:seeAlso	mrt:File

3.2 Ontological properties

The following Merritt ontological properties are defined.

mrt:hasPreview	
rdfs:label	hasPreview
rdfs:comment	Merritt property asserting that an object has a preview representation. The preview representation should present a concise summary of the object, such as a thumbnail for an image or an abstract for a document.
rdf:type	rdf:Property
rdfs:domain	mrt:Object
rdfs:range	mrt:File

mrt:isDerivativeOf	
rdfs:label	isDerivativeOf
rdfs:comment	Merritt property asserting that an object component is a derivative of another component. Note that both components must be of the same type. For example, a file can only be a derivative of another file.
rdf:type	rdf:Property
rdfs:domain	mrt:Component
rdfs:range	mrt:Component

mrt:metadataSchema	
rdfs:label	metadataSchema
rdfs:comment	Merritt property asserting that a file contains metadata conforming to a particular schema.
rdf:type	rdf:Property
rdfs:domain	mrt:File
rdfs:range	rdf:Literal

mrt:mimeType	
rdfs:label	mimeType
rdfs:comment	Merritt property asserting that a file has a particular MIME type. MIME types must either be registered in the IANA registry or be experimental types.
rdf:type	rdf:Property
rdfs:domain	mrt:File
rdfs:range	xsd:string

4 File system expression

Object components may originate from the object's producer, an object consumer, or be generated automatically by a Merritt service. The primary obligation of a Merritt curation environment is to ensure the long-term viability of producer-supplied content. There is a somewhat lower obligation regarding system-generated content, which presumably can be regenerated as necessary; and consumer-supplied content, which by definition originates outside of the object's primary curatorial context. The distinction between component provenance is explicitly represented in the file system representation of an object, c.f. [Dflat].

```
consumer/
        [ consumer-supplied-files
          ... ]
producer/
        [ producer-supplied-files
          ... ]
system/
          mrt-erc.txt                    # ERC metadata
        [ mrt-members.txt ]              # Class members
          mrt-membership.txt             # Collection membership
          mrt-mom.txt                    # MOM metadata
          mrt-object-map.ttl             # Resource map
          mrt-owner.txt                  # Owner class
        [ other-system-generated-files
          ... ]
```

References

[Abbott] Daisy Abbott, *What is Digital Curation?* April 3, 2008 <http://www.dcc.ac.uk/resource/briefing-papers/what-is-digital-curation/>.

[ARK] J. Kunze and R. Rodgers, *The ARK Identifer Schema*, May 22, 2008.

[Denning] Peter J. Denning, Chris Gunderson, and Rich Hayes-Roth, "Evolutionary system development," *Communications of the ACM* 51:17 (December 2008): 29-31.

[Dflat] UC3, *Dflat: A Simple File System Convention for Digital object Storage*, 2010.

[ERC] J. Kunze and A. Turner, *Kernel Metadata and Electronic Resource Citations*, April 28, 2009 <http://dublincore.org/kernelwiki/FrontPage?action=AttachFile&do=get&target=ercspec.html>.

[Fisher] David A. Fisher, *An Emergent Perspective on Interoperation in Systems of Systems*, CMU/SEI-2006-TR-003, ESC-TR-2006-003, March 2006 <http://www.sei.cmu.edu/pub/documents/06.reports/pdf/06tr003.pdf>.

[Foundations] UC3, *Digital Preservation Program: Foundations*, 2010.

[FRBR] IFLA Study Group on the Functional Requirements for Bibliographic Records, *Functional Requirements for Bibliographic Records – Final Report*, UBCIM 19 (February 2009; München: *K*. G. Saur, 1998).

[Ingest] UC3, *Merritt Ingest Service*, 2010.

[Inventory] UC3, *Merritt Inventory Service*, 2010.

[Merritt] UC3, *Merritt: An Emergent Approach to Digital Curation Infrastructure*, 2010.

[MIME] IANA, *MIME Media Types*, 2010 <http://www.iana.org/assignments/media-types>.

[ORE] Carl Lagoze, Herbert Van de Sompel, Pete Johnston, Michael Nelson, Robert Sanderson, and Simeon Warner, eds., *ORE User Guide – Primer*, October 17, 2008 <http://www.openarchives.org/ore/primer>.

[RDF] Graham Klyne and Jeremy J. Carroll, eds., *Resource Description Framework (RDF): Concepts and Abstract Syntax*, W3C Recommendation, February 10, 2004 <http://www.w3.org/TR/rdf-concepts/>.

[RFC2119] S. Bradner, *Key Words for Use in RFCs to Indicate Requirement Levels*, BCP 14, RFC 2119, March 1997 <http://www.ietf.org/rfc/rfc2119.txt>.

[Storage] UC3, *Merritt Storage Service*, 2010.

[Turtle] David Beckett and Tim Berners-Lee, *Turtle – Terse RDF Triple Language*, January 14, 2008 <http://www.w3.org/TeamSubmission/turtle/>.