UC3Merritt

University of California Curation Center

Merritt Replication Service

Rev. 1.0 — 2018-03-05

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. 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].

The Merritt Replication service manages the synchronization of content replicas across redundant storage locations to provide a globally-fault tolerant storage environment.

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 Replication Service

The Merritt Replication service is based on the following conceptual entities:

- Service
- Node
- Object
- Version
- File

2.1 Service

The initial conceptual entity is the Replication service itself, managing the synchronization of content replicas across redundant storage locations. The core service state properties SHALL consist of:

- Service name. [rpl:name]
- Service identifier, assigned to be globally unique among all UC3-controlled instantiations. [rpl:identifier]
- Service description. [rpl:description]
- Server status [rpl:status]
- Node set identifier. [rpl:nodeName]
- Thread pool size. [rpl:threadPool]
- Date of status report. [rpl:currentReportDate]
- Service specification and version. [rpl:serviceScheme]

2.1.1 Server status

A server status request SHALL return the core properties, plus the following REQUIRED additional properties:

- Replication processing flag: whether a replication operation is in progress. [rpl:replicationProcessing]
- Run replication flag: whether the replication service has been started. [rpl:runReplication]
- Status of the inventory database connection. [rpl:replicationSQL]
- Run count [rpl:cnt]
- Active replace-replica worker count [rpl:addQueueCnt]

2.1.2 Service state

A service state request SHALL return the core properties, plus the following REQUIRED additional properties:

- Timestamp of last replication [rpl:lastModified]

2.2 Node, Object, Version, and File

The Node, Object, Version, and File entities are identical to those in the Merritt Storage service [Storage], summarized below:

Node — A storage *node* is a digital object store, a system for managing some subset of the objects known to the service. Multiple nodes may be defined within the Storage service to represent different underlying storage technologies (e.g. disk vs. tape; FC vs. SATA), policy regimes (e.g. dark vs. bright), or other significant administrative categories.

NOTE In the Replication Service, nodes are additionally categorized as *primary* (access) or *secondary* (replication). Each Merritt collection is associated with one primary node and one or more secondary nodes. All objects in a collection are stored initially in the same primary node and replicated to the same set of secondary nodes.

Object — An *object* is a set of digitally encoded files whose change history is aggregated into discrete versions.

Version — A *version* is the particular configuration of an object at a point in time.

File — A *file* is a formatted digital manifestation of a unit of abstract content.

3 Service Interface

All Merritt services are defined in terms of abstract interfaces that can be implemented in various interactive modalities, including a procedural API with various language bindings, a command line API supported in various operating system command shells, and a RESTful API [Fielding].

State information about the various entities managed by the service MAY be requested in the following formats:

Format	***Extension***	***MIME type***
ANVL	`.txt`	`text/anvl`
JSON	`.json`	`application/json`
RDF/Turtle	`.ttl`	`text/turtle`
XHTML	`.html`	`application/xhtml+xml`

Format	*Extension*	*MIME type*
XML	`.xml`	`application/xml`

NOTE Until such time as a formal MIME types for the ANVL [ANVL] and Turtle [Turtle] formats are established at the IANA registry, the experimental MIME types "`text/x-anvl`" and "`text/x-turtle`" SHOULD be used, respectively.

4 Service Methods

The Replication service SHOULD support the following methods. Each method is first defined abstractly and then in terms of RESTful APIs.

NOTE The RESTful API is defined in terms of HTTP request and response messages. The notations "`UA`" and "`OS`" are used to distinguish the User Agent request from the Origin Server response. Names in *`italics`* indicate arbitrary, rather than fixed values. Brackets "`[`" and "`]`" enclose optional elements, parentheses "`(`" and "`)`" enclose groups of elements, a vertical bar "`|`" separates alternatives; and an ellipsis "**…**" indicates the arbitrary repetition of the previous element.

4.1 Get Server Status

Gets the status of the replication server. Used primarily for monitor functions to determine whether the server is up or down.

METHOD Get-status			[idempotent, safe]
—			No argument.
ResponseForm	Enum	Optional	Response form. The supported forms SHOULD include ANVL (default for command line interfaces), JSON, RDF/ Turtle, XHTML (default for web interfaces), and XML.
RETURN	Response form	Mandatory	Status of the replication server.
SIDE EFFECTS	—		
ERRORS	400	Badly-formed request.	
	401	Unauthorized user agent.	
	415	Unsupported response form.	
	503	Service unavailable.	
	500	Service error.	

- RESTful API

```
UA: GET /status[?t=form] HTTP/1.x
UA: Host: replic.cdlib.org
UA:

OS: HTTP/1.x 200 OK
OS: Content-type: response/form
OS:
OS: status
```

- Example:

```
curl -X GET 'http://<service>:38001/mrtreplic/status?t=xml'
```

4.2 Get Service State

Gets the state of the replication service. Because it includes a database query to get the timestamp of the last replication, this request can take several seconds.

METHOD Get-state			[idempotent, safe]
—			No argument.
ResponseForm	Enum	Optional	Response form. The supported forms SHOULD include ANVL (default for command line interfaces), JSON, RDF/ Turtle, XHTML (default for web interfaces), and XML.
RETURN	Response form	Mandatory	State of the replication service.
SIDE EFFECTS	—		
ERRORS	400	Badly-formed request.	
	401	Unauthorized user agent.	
	415	Unsupported response form.	
	503	Service unavailable.	
	500	Service error.	

- RESTful API

```
UA: GET /state[?t=form] HTTP/1.x
UA: Host: replic.cdlib.org
UA:

OS: HTTP/1.x 200 OK
OS: Content-type: response/form
```

```
OS:
OS: state
```

- Example:

```
curl -X GET 'http://<service>:38001/mrtreplic/state?t=xml'
```

4.3 Get File

Retrieves file content from the primary (access) node.

METHOD Get-file			[idempotent, safe]
Object	Identifier	Mandatory	Object primary identifier.
Version	Integer	Mandatory	Version number.
File	String	Mandatory	File identifier.
RETURN	Octet-stream	Mandatory	File content, retrieved from the access node.
SIDE EFFECTS	—		
ERRORS	400	Badly-formed request.	
	401	Unauthorized user agent.	
	404	Object, version, or file not found.	
	503	Service unavailable.	
	500	Service error.	

- RESTful API

```
UA: GET /content/object/version/file HTTP/1.x
UA: Host: replic.cdlib.org
UA:

OS: HTTP/1.x 200 OK
OS: Content-type: application/octet-stream
OS:
OS: content
```

- Example:

```
curl -X GET ' http://<service>:38001/mrtreplic/content/ark%3A%2F28722%2Fk23j3913
t/1/system%2Fmrt-ingest.txt'
```

4.4 Get Manifest

Retrieves the object XML manifest from the primary (access) node.

METHOD Get-manifest			[idempotent, safe]
Object	Identifier	Mandatory	Object primary identifier.
RETURN	Octet-stream	Mandatory	Object manifest, retrieved from the access node.
SIDE EFFECTS	—		
ERRORS	400	Badly-formed request.	
	401	Unauthorized user agent.	
	404	Object not found.	
	503	Service unavailable.	
	500	Service error.	

- RESTful API

```
UA: GET /manifest/object HTTP/1.x
UA: Host: replic.cdlib.org
UA:

OS: HTTP/1.x 200 OK
OS: Content-type: application/octet-stream
OS:
OS: manifest
```

- Example:

```
curl -X GET ' http://<service>:38001/mrtreplic/ manifest/ark%3A%2F28722%2Fk23j39
13t'
```

4.5 Set Server Status

Starts, pauses, or shuts down the server.

METHOD Set-service-state			[idempotent, unsafe]
Status	Enum	Optional	Server status: *start*, *pause*, *shutdown*.
ResponseForm	Enum	Optional	Response form. The supported forms SHOULD include ANVL (default for command line interfaces), JSON, RDF/ Turtle, XHTML (default for web interfaces), and XML.
RETURN	Response form	Mandatory	Server state.
SIDE EFFECTS	The service state is updated as specified.		

ERRORS	400	Badly-formed request.
	401	Unauthorized user agent.
	415	Unsupported response form.
	503	Service unavailable.
	500	Service error, or invalid status.

- RESTful API

```
UA: POST /service/[start|shutdown|pause] HTTP/1.x
UA: Host: replic.cdlib.org
UA:

OS: HTTP/1.x 200 OK
OS: Content-type: response/form
OS:
OS: state
```

- Example: starting the server

```
curl -X POST 'http://<service>:38001/mrtreplic/service/start?t=xml'
```

- Example: pausing the server

```
curl -X POST 'http://<service>:38001/mrtreplic/service/pause?t=xml'
```

- Example: shutting down the server

```
curl -X POST 'http://<service>:38001/mrtreplic/service/shutdown?t=xml'
```

4.6 Delete inventory records

Deletes the replica database records (but not replicated content), including audit records, for a replicated object. Primary records are not affected.

METHOD Delete-inventory-records			[idempotent, unsafe]
Node	Identifier	Mandatory	Node identifier.
Object	Identifier	Mandatory	Object primary identifier.
ResponseForm	Enum	Optional	Response form. The supported forms SHOULD include ANVL (default for command line interfaces), JSON, RDF/ Turtle, XHTML (default for web interfaces), and XML.
RETURN	Response form	Mandatory	Result of the delete operation.

SIDE EFFECTS	Deletes the inventory database records (but not stored content) for the specified object on the specified secondary storage node. It is an error to call this method with the ID of a primary storage node.	
ERRORS	400	Badly-formed request.
	401	Unauthorized user agent.
	404	Node or object not found.
	415	Unsupported response form.
	503	Service unavailable.
	500	Service error.

- RESTful API

```
UA: DELETE /invdelete/node/object HTTP/1.x
UA: Host: replic.cdlib.org
UA:

OS: HTTP/1.x 200 OK
OS: Content-type: response/form
OS:
OS: result
```

- Example:

```
curl -X DELETE ' http://<service>:38001/mrtreplic/ ark%3A%2F28722%2Fk23j3913t?t=
xml'
```

4.7 Delete secondary content and records

Deletes the inventory database records and stored content for all replicas of the specified object. Primary records and content are not affected.

METHOD Delete-secondary-content			[idempotent, unsafe]
Object	Identifier	Mandatory	Object primary identifier.
ResponseForm	Enum	Optional	Response form. The supported forms SHOULD include ANVL (default for command line interfaces), JSON, RDF/ Turtle, XHTML (default for web interfaces), and XML.
RETURN	Response form	Mandatory	Result of the delete operation.
SIDE EFFECTS	Deletes all secondary content (file replicas) and related database records for the specified object.		
ERRORS	400	Badly-formed request.	
	401	Unauthorized user agent.	
	404	Node or object not found.	

	415	Unsupported response form.
	503	Service unavailable.
	500	Service error.

- RESTful API

```
UA: DELETE /deletesecondary/object HTTP/1.x
UA: Host: replic.cdlib.org
UA:

OS: HTTP/1.x 200 OK
OS: Content-type: response/form
OS:
OS: result
```

- Example:

```
curl -X DELETE 'http://<service>:38001/mrtreplic/deletesecondary/ark%3A%2F28722%
2Fk23j3913t?t=xml'
```

4.8 Add replicas

Triggers the replication of a specified object, bypassing the replication queue. (In ordinary operation, objects to be replicated are identified automatically on a rolling basis by querying the database.)

METHOD Add-replicas			[idempotent, unsafe]
Object	Identifier	Mandatory	Object primary identifier.
ResponseForm	Enum	Optional	Response form. The supported forms SHOULD include ANVL (default for command line interfaces), JSON, RDF/ Turtle, XHTML (default for web interfaces), and XML.
RETURN	Response form	Mandatory	Result of the operation.
SIDE EFFECTS	Replicates the specified object to the appropriate secondary nodes. Used for testing and for correcting errors due to incomplete processing.		
ERRORS	400	Badly-formed request.	
	401	Unauthorized user agent.	
	404	Object not found.	
	415	Unsupported response form.	
	503	Service unavailable.	
	500	Service error.	

- RESTful API

```
UA: POST /add/object HTTP/1.x
UA: Host: replic.cdlib.org
UA:

OS: HTTP/1.x 200 OK
OS: Content-type: response/form
OS:
OS: result
```

- Example:

```
curl -X POST 'http://<service>:38001/mrtreplic/add/ark%3A%2F28722%2Fk23j3913t?t=
xml'
```

4.9 Add inventory record

Confirms content replication and recreates inventory records for the specified object.

METHOD Add-inventory-record			[idempotent, unsafe]
Object	Identifier	Mandatory	Object primary identifier.
ResponseForm	Enum	Optional	Response form. The supported forms SHOULD include ANVL (default for command line interfaces), JSON, RDF/ Turtle, XHTML (default for web interfaces), and XML.
RETURN	Response form	Mandatory	Result of the operation.
SIDE EFFECTS	Recreates the replication records for the specified object in the inventory database. Used for testing and for correcting errors due to incomplete processing.		
ERRORS	400	Badly-formed request.	
	401	Unauthorized user agent.	
	404	Object not found.	
	415	Unsupported response form.	
	503	Service unavailable.	
	500	Service error.	

- RESTful API

```
UA: POST /addinv/object HTTP/1.x
UA: Host: replic.cdlib.org
UA:
```

```
OS: HTTP/1.x 200 OK
OS: Content-type: response/form
OS:
OS: result
```

- Example:

```
curl -X POST 'http://<service>:38001/mrtreplic/addinv/ark%3A%2F28722%2Fk23j3913t
?t=xml'
```

4.10 Add collection-to-node mapping

Maps the specified collection to the specified node. Content in that collection will be queued for replication to that node.

METHOD Add-mapping			[idempotent, unsafe]
Collection	Identifier	Mandatory	Collection identifier.
Node	Identifier	Mandatory	Node identifier.
ResponseForm	Enum	Optional	Response form. The supported forms SHOULD include ANVL (default for command line interfaces), JSON, RDF/ Turtle, XHTML (default for web interfaces), and XML.
RETURN	Response form	Mandatory	The result of the mapping operation.
SIDE EFFECTS	Adds a mapping for the specified collection to the specified storage node.		
ERRORS	400	Badly-formed request.	
	401	Unauthorized user agent.	
	404	Collection or node not found.	
	415	Unsupported response form.	
	503	Service unavailable.	
	500	Service error.	

- RESTful API

```
UA: POST /addmap/collection/node HTTP/1.x
UA: Host: replic.cdlib.org
UA:

OS: HTTP/1.x 200 OK
OS: Content-type: response/form
OS:
OS: result
```

- Example:

```
curl -X POST 'http://<service>:38001/mrtreplic/
ark%3A%2F13030%2Fm52b8w1p/5001?t=xml'
```

4.11 Match replicas across nodes

Checks whether the manifests and inventory records for the specified object match across the specified nodes.

METHOD Match-replicas			[idempotent, safe]
Source Node	Identifier	Mandatory	Source node identifier.
Target Node	Identifier	Mandatory	Target node identifier.
Object	Identifier	Mandatory	Object primary identifier.
ResponseForm	Enum	Optional	Response form. The supported forms SHOULD include ANVL (default for command line interfaces), JSON, RDF/ Turtle, XHTML (default for web interfaces), and XML.
RETURN	Response form	Mandatory	Match results.
SIDE EFFECTS	—		
ERRORS	400	Badly-formed request.	
	401	Unauthorized user agent.	
	404	Node or object not found.	
	415	Unsupported response form.	
	503	Service unavailable.	
	500	Service error.	

- RESTful API

```
UA: GET /match/sourceNode/targetNode/object HTTP/1.x
UA: Host: replic.cdlib.org
UA:

OS: HTTP/1.x 200 OK
OS: Content-type: response/form
OS:
```

- Example:

```
curl -X GET 'http://<service>:38001/mrtreplic/match/2111/5001/ark%3A%2F28722%2Fk
2v11vk9d?t=xml'
```

4.12 Match manifest to storage node

Confirms that the manifest for the specified object stored on the specified node matches the inventory database records for the specified object.

METHOD Match-manifest			[idempotent, safe]
Source Node	Identifier	Mandatory	Node identifier.
Object	Identifier	Mandatory	Object primary identifier.
ResponseForm	Enum	Optional	Response form. The supported forms SHOULD include ANVL (default for command line interfaces), JSON, RDF/ Turtle, XHTML (default for web interfaces), and XML.
RETURN	Response form	Mandatory	Match results.
SIDE EFFECTS	—		
ERRORS	400	Badly-formed request.	
	401	Unauthorized user agent.	
	404	Node or object not found.	
	415	Unsupported response form.	
	503	Service unavailable.	
	500	Service error.	

- RESTful API

```
UA: GET /match/sourceNode/object HTTP/1.x
UA: Host: replic.cdlib.org
UA:

OS: HTTP/1.x 200 OK
OS: Content-type: response/form
OS:
```

- Example:

```
curl -X GET 'http://<service>:38001/mrtreplic/match/2111/ark%3A%2F28722%2Fk2v11v
k9d?t=xml'
```

References

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

[ANVL] John Kunze, Brewster Kahle, Julien Masanès, Gordon Mohr, *A Name-Value Language (ANVL)*, August 28, 2005. <https://tools.ietf.org/html/draft-kunze-anvl-02>

[Denning] Peter J. Denning, Chris Gunderson, and Rich Hayes-Roth, "Evolutionary system development," *Communications of the ACM* 51:12 (December 2008): 29-31 <https://doi.org/10.1145/1409360.1409371>

[Fielding] Roy Fielding and Richard Taylor, "Principled design of the modern web architecture," *ACM Transactions on Internet Technology* 2:2 (May 2002): 115-150 <https://doi.org/10.1145/514183.514185>.

[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>.

[Merritt] Stephen Abrams, John Kunze, and David Loy, "An Emergent Micro-Services Approach to Digital Curation Infrastructure", *International Journal of Digital Curation* 5:1 (2010) <https://doi.org/10.2218/ijdc.v5i1.151>.

[RFC 2119] 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] Stephen Abrams, John Kunze, and David Loy, "Merritt Storage Service", <http://n2t.net/ark:/13030/m54f6mfn>

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