# University of California Curation Center

**Merritt Audit Service**

Rev 1.0 – 2013-02-04

# 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 Audit service manages the ongoing bit-level verification of digital content. This content MAY be managed in a Merritt curation environment or in non-Merritt systems and services.

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 Requirements

The Audit service is used to verify the bit-level integrity of digital content. The service MUST meet the following functional and non-function requirements:

* Fixity tests (initially) SHALL be limited to determining whether bit-level data corruption has occurred. This SHALL be a two stage determination:
  + Fixity testing SHALL first verify the size of the current content relative to a stored veridical value.
  + If necessary, fixity testing SHALL then verify the message digest of the current

content with relative to a stored veridical value.

NOTE Any discrepancy in content size obviates the need for further verification of the digest since, as a matter of explicit algorithmic intent, it should always differ from the stored value.

* + The Audit service SHOULD support the following digest types:
    - Adler-32, CRC-32
    - MD2, MD5
    - SHA-1, SHA-256, SHA-384, SHA-512
  + The service SHALL support the following distinguishable status types for a given content item:
    - *Unverified*. No attempt has yet been made to verify the item.
    - *Verified* The item has been fully verified with respect to size and digest.
    - *Size mismatch*. The item does not match its veridical size.
    - *Digest mismatch*. The item does not match its veridical message digest value.
    - *Unavailable*. The item cannot be retrieved.
    - *Processing*. The item is currently being processed by the Audit microservice.

* Fixity testing SHALL be performed on an iterative basis for all content items known to the service.
  + The iteration SHALL be in the temporal order of the date/timestamp of the last fixity test for the items, from the oldest to the most recent date/timestamp.
  + A fixity date/time threshold MAY be supported (e.g. test items that have not been tested in the last 3 months). If date threshold is supported it WILL apply to all fixity content (e.g. no content specific threshold will exist).
  + A summary status report SHALL be sent to the notification email at the conclusion of each iteration.

* The service MUST support the addition of new content items.
  + An item MUST include a veridical size and message digest value.
  + An item MUST include a unique location.
    - This location MAY be either an http- or file-scheme URL.
    - File-scheme URLs MUST specify an absolute file pathname, resolvable to a locally-mounted file system.
  + An item MAY include a contextual label that can be used for contextual searching of items.
  + An item MAY include a descriptive note. o An item MAY be updated or deleted.

* Only the Audit service provider SHALL receive notification of fixity errors uncovered during iterative processing; there is no requirement that the actual content owners or curators receive any direct notification from the service. This notification SHALL be provided on a periteration basis, and SHALL include the set of all items whose status is not *verified*.

* The service MUST support requests for changing the status of its processing, permitting graceful transitions between *running*, *paused*, and *shutdown* states.

|  |  |  |  |
| --- | --- | --- | --- |
| ***Status*** | ***Add/update/delete items Verify items*** | | ***Get service/item state*** |
| *Running* | Available | Available | Available |
| *Paused* | Unavailable | Available | Available |
| *Shutdown* | Unavailable | Unavailable | Unavailable |

* The service MUST scale gracefully to ten million items.
  + Multi-threading MAY be used to optimize performance.

# 3 Audit Service

The Merritt Audit service is based on the following conceptual entities, each defined in terms of its specific state properties.

* Service
* Item
* Items report
* SQL report
* Submission

## 3.1 Service

The initial conceptual entity is the Audit service itself. The global service state properties MUST minimally include:

|  |  |  |
| --- | --- | --- |
| • | Service name. | [fix:name] |
| • | Service identifier, assigned to be globally unique among all UC3-controlled instantiations. | [fix:identifier] |
| • | Service description. | [fix:description] |
| • | Service implementation version. | [fix:version] |
| • | Iteration interval, in days. 0 indicates that iterations run continuously. | [fix:interval] |
| • | Thread pool size. | [fix:threadPool] |
| • | Last iteration date/timestamp. | [fix:lastIteration] |
| • | Last iteration elapsed time. | [fix:elapsedTime] |
| • | Queue sleep, number of seconds between fixity tests | [fix:queueSleep] |

* Total size, in octets, of all items. [fix:totalSize]
* Number of items. [fix:numItems]
* Number of unverified items. [fix:numUnverified]
* Number of failed items. [fix:numFailedItems]
* Number of unavailable items. [fix:numUnavailable]
* Service status: *paused*, *running*, *shutdown*. [fix:status]
* Creation date/timestamp. [fix:created]
* Modification date/timestamp. [fix:lastModified]
* Service specification and version. [fix:serviceScheme]
* Base URI for service method invocations. [fix:baseURI]
* Notification email. [fix:notification]
* Customer support URI. [fix:supportURI]
* Periodic report frequency, number of days between periodic [fix:periodicReqportFrequency] reports. The report is mailed to the notification email address.
* Periodic report format, format for fixity state report as [fix:periodicReportFormat] attached to email.

The notification threshold is the limit to the number of individual item notifications that will be sent during a given iteration.

## 3.2 Item

An *item* is a discrete unit of digital content uniquely identified and located by an http- or file-scheme URL. The item state properties MUST minimally include:

* Item URL. [item:URL]
* Item source: *merritt*, *file*, or *web*. [item:source]
* Veridical size, in octets. [item:size]
* Last size, in octets. [item:lastSize]
* Message digest type: *Adler-32*, *CRC-32*, *MD2*, *MD5*, *SHA-1*, [item:digestType] *SHA-256*, *SHA-384*, or *SHA-512*.
* Veridical message digest value, as a hexadecimal string. [item:digestValue]
* Last message digest value, as a hexadecimal string. [item:lastDigestValue]
* Last verification status: *unverified*, *in-process*, *verified*, [item:status] *size-mismatch*, *digest-mismatch*, *unavailable*.
* Last verification date/timestamp. [item:verified]
* Item contexts. [item:contexts] o Item context. [item:context] • Descriptive note. [item:note]
* Creation date/timestamp. [item:created]
* Modification date/timestamp. [item:modified]

The source indicates the managerial context in which the item is located. Items sourced in Merritt

MAY be verified remotely by Merritt, rather than locally by the service, if the service mode is *remote*.

An item’s status is:

* *Unverified* if the item has never been subject to an iterative verification;
* *In-process* if the iterative verification for the item is in process;
* *Verified* if the iterative verification determines that the current size and message digest value match exactly the veridical size and digest;
* *Size-mismatch* if the item’s current size is different from the veridical size;
* *Digest-mismatch* if the item’s current digest value is different from the veridical value; and
* *Unavailable* if the item cannot be retrieved using its URL.

The last size and digest value SHALL be *null* if the item status is *unverified*, and SHALL match exactly the veridical size and digest value if the status is *verified*. The last size and digest value SHOULD differ from the veridical size and value if the status is *size-mismatch*. The last size MAY, and the digest value SHOULD, differ from the veridical size and value if the status is *digest-mismatch*.

The item context is supported to permit contextual searching of items. It is assumed that the context string is hierarchical in the left-to-right direction so that suffix wildcard matching can be used to select meaningfully-related sets of items at appropriate levels of granularity. The context is intended for automated processing; the descriptive note, on the other hand, is intended for human consumption. An item may be associated with an arbitrary number of contexts.

NOTE For object files managed in a Merritt repository, separate contexts will be defined for each file’s structural signature, that is, the unique concatenation of node/object/version/file identifiers, and for each collection of which the file’s object is a member.

## 3.3 Items Report

A collection of one or more Items as a result of a query

|  |  |  |
| --- | --- | --- |
| • | Creation date/timestamp. | [items:created] |
| • | Report query type. | [items:type] |
| • | Context | [items:context] |
| • | Items | [items:items] |
|  | o Item entry.   [item state content] | [items:item] |

The report query type and context are the values used in the request report.

## 3.4 SQL Report

A collection of one or more sets of cell values returned as part of an SQL SELECT.

|  |  |  |
| --- | --- | --- |
| • | Creation date/timestamp. | [fixsql:created] |
| • | Select. | [fixsql:type] |
| • | Rows | [fixsql:rows] |
|  | o Row.   [key-values returned from select] | [fixsql:row] |

The report query type and context are the values used in the request report.

## 3.5 Submission

The submission state indicates whether a report delivered by email was accepted or not.

|  |  |  |
| --- | --- | --- |
| • | Submission date/timestamp. | [fixsub:dateSubmitted] |
| • | Submision status: *true* or *false*. | [fixsub:isSubmitted] |

# 4 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 |
| CSV | .csv | text/csv |
| HTML | .html | application/xhtml+xml |
| JSON | .json | application/json |
| RDF/Turtle | .ttl | text/turtle |
| XML | .xml | application/xml |

Table 1 – State formats

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/xturtle” SHOULD be used, respectively.

# 5 Service Methods

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

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.

## 5.1 Help

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| METHOD Help | [*idempotent*, *safe*] | | | |
| Method | Enum | | Optional | Specific method about which help is requested. |
| 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 | ResponseForm | | Mandatory | Help information about the specific method or the service as a whole. |
| 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 /help[?t=*form*] HTTP/1.*x*

UA: Host: *audit*.cdlib.org

UA: Accept: *response/form* UA:

OS: HTTP/1.*x* 200 OK

OS: Content-type: *response/form*

OS:

OS: *help*

## 5.2 Get Service State

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD Get-service-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 | Service state. |
| 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*: audit*.cdlib.org UA:

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

OS: *state*

## 5.3 Get Item State

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD Get-item-state [*idempotent*, *safe*] | | | |
| URL | URL | Mandatory | Item URL. |
| 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 | Item Status. |
| 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/*url*[?t=form] HTTP/1.*x* UA:Host*: audit*.cdlib.org UA:

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

OS: *state*

## 5.4 Add Item

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD Add-item [*non-idempotent*, *unsafe*] | | | |
| URL | URL | Mandatory | URL for the item, using the “http” or “file” schemes. File URLs MUST represent absolute pathnames. |
| Source | Enum | Mandatory | Source: *merritt*, *file*, or *web*. |
| Size | Integer | Mandatory | Veridical size, in octets. |
| Digest-type | Enum | Mandatory | Message digest type: *Adler-32*, *crc-32*, *md2*, *md5*, *sha-1*, *sha-256*, *shaA-384,* or  *sha-512* . |
| Digest-value | String | Mandatory | Veridical message digest value, as a hexadecimal string. |
| Context | String | Optional | User defined element supporting contextual search. |
| Note | String | Optional | Descriptive note. |
| 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 | Item state |
| SIDE EFFECTS | A new item is added to the service,with the created date/timestamp set to the current value, and is immediately verified. | | |
| ERRORS | 400 Badly-formed request | | |
| 401 Unauthorized user agent. | | |
| 404 Item not found. | | |
| 415 Unsupported response form. | | |
| 503 Service unavailable. | | |
| 500 Service error. | | |
| NOTE | Although context is a repeatable element of the item state, this method only permits a single context to be specified. If multiple contexts are desired, the *Update*-*item* method MUST be used.    Features: fixity test; add to fixity database | | |

• RESTful API

NOTE RESTful requests are formatted assuming an underlying HTML form [HTML] and using the “multipart-form-data” content type [Multipart].

UA: POST /add HTTP/1.*x*

UA: Host: *audit*.cdlib.org

UA: Content-type: multipart/form-data; boundary=*boundary*

UA:

UA: --*boundary*

UA:

UA: adler-32 | crc-32 | md2 | md5 | sha-1 | sha-256 | sha-384 | sha-512

UA: --*boundary*

UA: Content-disposition: form-data; name=”digest-value”

UA:

UA: *value*

UA: --*boundary* [

UA: Content-disposition: form-data; name=”context”

UA:

UA: *context*

UA: ...

UA: *--boundary* ]

UA: Content-disposition: form-data; name=”note”

UA:

UA: *note*

UA: --*boundary* ] [

UA: Content-disposition: form-data; name=”response-form”

UA:

UA: *form*

UA: --*boundary* ]

OS: HTTP/1.*x* 201 Created

OS: Content-type: *response/form*

OS: Location: http://*audit*.cdlib.org/state/*url* OS:

OS: *state*

## 5.5 Queue Item

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD Queue-item [*non-idempotent*, *unsafe*] | | | |
| URL | URL | Mandatory | URL for the item, using the “http” or “file” schemes. File URLs MUST represent absolute pathnames. |
| Source | Enum | Mandatory | Source: *merritt*, *file*, or *web*. |
| Size | Integer | Mandatory | Veridical size, in octets. |
| Digest-type | Enum | Mandatory | Message digest type: *Adler-32*, *crc-32*, *md2*, *md5*, *sha-1*, *sha-256*, *shaA-384,* or  *sha-512* . |
| Digest-value | String | Mandatory | Veridical message digest value, as a hexadecimal string. |
| Context | String | Optional | User defined element supporting contextual search. |
| Note | String | Optional | Descriptive note. |
| 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 | Item state |
| SIDE EFFECTS | A new item is added to the service,with the created date/timestamp set to the current value, and is immediately verified. | | |
| ERRORS | 400 Badly-formed request | | |
| 401 Unauthorized user agent. | | |
| 404 Item not found. | | |
| 415 Unsupported response form. | | |
| 503 Service unavailable. | | |
| 500 Service error. | | |
| NOTE | Although context is a repeatable element of the item state, this method only permits a single context to be specified. If multiple contexts are desired, the *Update*-*item* method MUST be used.    Features: no fixity test; add to fixity database | | |

• RESTful API

NOTE RESTful requests are formatted assuming an underlying HTML form [HTML] and using the “multipart-form-data” content type [Multipart].

UA: POST /queue HTTP/1.*x*

UA: Host: *audit*.cdlib.org

UA: Content-type: multipart/form-data; boundary=*boundary*

UA:

UA: --*boundary*

UA:

UA: adler-32 | crc-32 | md2 | md5 | sha-1 | sha-256 | sha-384 | sha-512

UA: --*boundary*

UA: Content-disposition: form-data; name=”digest-value”

UA:

UA: *value*

UA: --*boundary* [

UA: Content-disposition: form-data; name=”context”

UA:

UA: *context*

UA: ...

UA: *--boundary* ]

UA: Content-disposition: form-data; name=”note”

UA:

UA: *note*

UA: --*boundary* ] [

UA: Content-disposition: form-data; name=”response-form”

UA:

UA: *form*

UA: --*boundary* ]

OS: HTTP/1.*x* 201 Created

OS: Content-type: *response/form*

OS: Location: http://*audit*.cdlib.org/state/*url*

OS:

OS: *state*

## 5.6 Test Item

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD Test-item [*idempotent*, *safe*] | | | |
| URL | URL | Mandatory | URL for the item, using the “http” or “file” schemes. File URLs MUST represent absolute pathnames. |
| Source | Enum | Mandatory | Source: *merritt*, *file*, or *web*. |
| Size | Integer | Mandatory | Veridical size, in octets. |
| Digest-type | Enum | Mandatory | Message digest type: *Adler-32*, *crc-32*, *md2*, *md5*, *sha-1*, *sha-256*, *shaA-384,* or  *sha-512* . |
| Digest-value | String | Mandatory | Veridical message digest value, as a hexadecimal string. |
| Context | String | Optional | User defined element supporting contextual search. |
| Note | String | Optional | Descriptive note. |
| 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 | Item state |
| SIDE EFFECTS | A new item is added to the service,with the created date/timestamp set to the current value, and is immediately verified. | | |
| ERRORS | 400 Badly-formed request | | |
| 401 Unauthorized user agent. | | |
| 404 Item not found. | | |
| 415 Unsupported response form. | | |
| 503 Service unavailable. | | |
| 500 Service error. | | |
| NOTE | Although context is a repeatable element of the item state, this method only permits a single context to be specified. If multiple contexts are desired, the *Update*-*item* method MUST be used.    Features: fixity test; no add to fixity database | | |

• RESTful API

NOTE RESTful requests are formatted assuming an underlying HTML form [HTML] and using the “multipart-form-data” content type [Multipart].

UA: POST /test HTTP/1.*x*

UA: Host: *audit*.cdlib.org

UA: Content-type: multipart/form-data; boundary=*boundary*

UA:

UA: --*boundary*

UA:

UA: adler-32 | crc-32 | md2 | md5 | sha-1 | sha-256 | sha-384 | sha-512

UA: --*boundary*

UA: Content-disposition: form-data; name=”digest-value”

UA:

UA: *value*

UA: --*boundary* [

UA: Content-disposition: form-data; name=”context”

UA:

UA: *context* UA: ...

UA: *--boundary* ]

UA: Content-disposition: form-data; name=”note”

UA:

UA: *note*

UA: --*boundary* ] [

UA: Content-disposition: form-data; name=”response-form”

UA:

UA: *form*

UA: --*boundary* ]

OS: HTTP/1.*x* 201 Created

OS: Content-type: *response/form*

OS: Location: http://*audit*.cdlib.org/state/*url*

OS:

OS: *state*

### 5.7 Update Item

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD Update-item [*idempotent*, *unsafe*] | | | |
| URL | URL | Mandatory | URL for the item, using the “http” or “file” schemes. File URLs MUST represent absolute pathnames. |
| Source | Enum | Optional | Source: *merritt*, *file*, or *web*. |
| Size | Integer | Optional | Veridical size, in octets. |
| Digest-type | Enum | Optional | Message digest type: *adler-32*, *crc-32*, *md2*, *md5*, *sha-1*, *sha-25,* *sha-384*, or *sha-512*. |
| Digest-value | String | Veridical message digest value, as a hexadecimal string. |
| Context | String | Optional | User defined element supporting contextual search. |
| Note | String | Optional | Descriptive note. |
| 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 | Item state |
| SIDE EFFECTS | Item state is updated, with the status set to *unverified*, last size and digest value and verification date set to *null*, and the last modified date/timestamp set to the current value. | | |
| ERRORS | 400 Badly-formed request | | |
| 401 Unauthorized user agent. | | |
| 404 Item not found. | | |
| 415 Unsupported response form. | | |
| 503 Service unavailable. | | |
| 500 Service error. | | |
| NOTE | Although context is a repeatable element of the item state, this method only permits a single context to be specified. If multiple contexts are desired, the method MUST be invoked repeatedly. | | |

• RESTful API

UA: POST /update HTTP/1.*x* UA: Host: *audit*.cdlib.org

UA: Content-type: multipart/form-data; boundary=*boundary*

UA:

UA: --*boundary*

UA: Content-disposition: form-data; name=”url”

UA:

UA: *url* UA: --*boundary* [

UA: Content-disposition: form-data; name=”source”

UA:

UA: merritt | file | web

UA: --*boundary* ] [

UA: Content-disposition: form-data; name=”size”

UA:

UA: *size*

UA: --*boundary* ] [

UA: Content-disposition: form-data; name=”digest-type”

UA:

UA: adler-32 | crc-32 | md2 | md5 | sha-1 | sha-256 | sha-384 | sha-512

UA: --*boundary*

UA: Content-disposition: form-data; name=”digest-value”

UA:

UA: *value*

UA: --*boundary* ][

UA: Content-disposition: form-data; name=”context”

UA:

UA: *context*

UA: ...

UA: *--boundary* ]

UA: Content-disposition: form-data; name=”note”

UA:

UA: *note*

UA: --*boundary* ] [

UA: Content-disposition: form-data; name=”response-form”

UA:

UA: *form*

UA: --*boundary* ]

OS: HTTP/1.*x* 200 OK

OS: Content-type: *response/form* OS:

OS: *state*

### 5.8 Delete Item

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD Delete-item [*idempotent*, *unsafe*] | | | |
| URL | URL | Mandatory | URL for the item. |
| 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 | Item state |
| SIDE EFFECTS | Item is deleted from the service. | | |
| ERRORS | 400 Badly-formed request | | |
| 401 Unauthorized user agent. | | |
| 404 Item not found. | | |
| 415 Unsupported response form. | | |
| 503 Service unavailable. | | |
| 500 Service error. | | |

• RESTful API

UA: DELETE /item/*url*[?t=*form*] HTTP/1.*x*

UA: Host: *audit*.cdlib.org UA: Accept: *response*/*form* UA:

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

OS: *state*

## 5.9 Set Service State

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD Set-service-state [*idempotent*, *unsafe*] | | | |
| Status | Enum | Optional | Service status: *pause*, *resume*, *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 | Service state |
| SIDE EFFECTS | The service status is updated as specified. | | |
| ERRORS | 400 Badly-formed request. | | |
| 401 Unauthorized user agent. | | |
| 415 Unsupported response form. | | |
| 503 Service unavailable. | | |
| 500 Service error. | | |

• POST API

UA: POST /service/[pause|resume|shutdown][&t=*form*] HTTP/1.*x*

UA:Host*: audit*.cdlib.org UA: Accept: *response/form* UA:

OS: HTTP/1.*x* 200 OK

OS: Content-type: *response/form*

OS:

OS: *state*

## 5.10 Request Item Report

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD Request-report [*idempotent*, *safe*] | | | |
| Type | Enum | Mandatory | Report type: *all*, *failed*, *ad-hoc*. |
| Context | String | Optional | Item context, possibly including a trailing wildcard (defaults to the most inclusive wildcard). |
|  |  |  |  |
| Email | String | Optional | Email address to which the report is sent (defaults to the service notification email address). |
| Response form | Enum | Optional | Response form. The supported forms SHOULD include ANVL, CSV (default), JSON, RDF/Turtle, XHTML, and XML. |
| RETURN | Response form | Mandatory | Items Report |
| SIDE EFFECTS | An immediate synchronous response is sent containing the request parameters. The requested report is queued for asynchronous delivery. The reporting set is formed by all items matching the context string. All items in the set are reported for *all* reports; only those items whose status is *size-mismatch* or *digest-mismatch* are reported for *failed* reports; and only those items matching the ad-hoc query are reported for *ad-hoc* reports.  All reports SHALL include the service state, the request parameters, and the item state for all reportable items. | | |
| ERRORS | 400 Badly-formed request. | | |
| 401 Unauthorized user agent. | | |
| 415 Unsupported response form. | | |
| 503 Service unavailable. | | |
| 500 Service error. | | |
| NOTE | Care must be exercised in examining and sanitizing, if necessary, the query string to guard against malicious injection exploits. | | |

• RESTful API

UA: POST /report HTTP/1.*x* UA: Host: *audit*.cdlib.org

UA: Content-type: multipart/form-data; boundary=*boundary*

UA:

UA: --*boundary*

UA: Content-disposition: multipart/form-data; name=”type”

UA:

UA: all | failed | ad-hoc

UA: --*boundary* [

UA: Content-disposition: form-data; name=”context”

UA:

UA: *context* ] [

UA: --*boundary*

UA:Content-disposition: form-data; name=”query”

UA:

UA: *query*

UA: --*boundary* ] [

UA: Content-disposition: form-data; name=”email” UA:

UA: *email*

UA: *--boundary* ] [

UA: Content-disposition: form-data; name=”response-form”

UA:

UA: *form*

UA: --*boundary* ]

OS: HTTP/1.*x* 200 OK

OS: Content-type: *response/form*

OS: request

## 5.11 Request SQL Report

|  |  |  |  |
| --- | --- | --- | --- |
| METHOD Request-SQL-report [*idempotent*, *safe*] | | | |
| Select | String | Optional | Implementation-specific query for ad-hoc queries. |
| Email | String | Optional | Email address to which the report is sent (defaults to the service notification email address). |
| Response form | Enum | Optional | Response form. The supported forms SHOULD include ANVL, CSV (default), JSON, RDF/Turtle, XHTML, and XML. |
| RETURN | Response form | Mandatory | SQL Report. |
| SIDE EFFECTS | An immediate synchronous response is sent containing the request parameters. The requested report is queued for asynchronous delivery. The reporting set is formed by all items matching the context string. All items in the set are reported for *all* reports; only those items whose status is *size-mismatch* or *digest-mismatch* are reported for *failed* reports; and only those items matching the ad-hoc query are reported for *ad-hoc* reports.  All reports SHALL include the service state, the request parameters, and the item state for all reportable items. | | |
| ERRORS | 400 Badly-formed request. | | |
| 401 Unauthorized user agent. | | |
| 415 Unsupported response form. | | |
| 503 Service unavailable. | | |
| 500 Service error. | | |
| NOTE | Care must be exercised in examining and sanitizing, if necessary, the query string to guard against malicious injection exploits. | | |

• RESTful API

UA: POST /select HTTP/1.*x* UA: Host: *audit*.cdlib.org

UA: Content-type: multipart/form-data; boundary=*boundary*

UA:

UA: --*boundary*

UA: Content-disposition: multipart/form-data; name=”select”

UA:

UA: status,count(status) from fx\_item group by status

UA: --*boundary* [

UA: Content-disposition: form-data; name=”email”

UA:

UA: *email*

UA: *--boundary* ] [

UA: Content-disposition: form-data; name=”response-form”

UA:

UA: *form*

UA: --*boundary* ]

OS: HTTP/1.*x* 200 OK

OS: Content-type: *response/form* OS:

OS: *request*

# 6 Implementation

## 6.1 Audit Home

The Audit service is instantiated in a file system with the following structure:

<*audit\_home*>/

0=audit\_1.0 admin/ log/ rewrite.txt (optional) audit-info.txt

The REQUIRED file “audit-info.txt” declares the global Audit service properties, for example:

name: UC3 description: UC3 fixity micro-service interval: 0 threadPool: 5 queueSleep: 0 serviceScheme: Fixity/0.2/1.0 baseURI: http://*audit*.cdlib.org/ notificationEmail: mailto:audit-support@ucop.edu supportURI: mailto:uc3-support@ucop.edu periodicReportFrequency: 24 auditQualify: and id%4 = 0 nodeNode=nodes-prod

For the OPTIONAL “rewrite.txt” file, see below.

## 6.2 SQL

Due to the need for efficient queries on 10s of millions of items, the Audit service uses an embedded SQL RDBMS to manage the items and item contexts.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Item table*** |  |  |  |  |  |
| ***Name*** | ***Type*** | ***Key*** | ***Null?*** | ***Indexed?*** | ***Value*** |
| itemkey | Integer | PK | No | Yes | Primary key (auto-assigned from a sequence). |
| url | Varchar |  | No | Yes | Item URL. |
| source | Varchar |  | No | Yes | Item source: *merritt*, *file*, or *web*. |
| size | Integer |  | No | No | Veridical size, in octets. |
| type | Varchar |  | No | Yes | Message digest type: *adler-32*, *crc-32*, *md2*, *md5*, *sha-1*, *sha-256*, *sha-384*, or *sha-512*. |
| value | Varchar |  | No | No | Veridical message digest value, in hexadecimal. |
| status | Varchar |  | No | Yes | Verification status: *unverified*, *verified*, *in-* |
|  |  |  |  |  | *process*, *size-mismatch*, *digest-mismatch*, or *unavailable*. |
| verified | Date |  | Yes | Yes | Verification date/timestamp. |
| lastsize | Integer |  | Yes | No | Last size, in octets. |
| lastvalue | Varchar |  | Yes | No | Last digest value, in hexadecimal. |
| note | Varchar |  | Yes | No | Descriptive note. |
| created | Date |  | No | Yes | Item creation date/timestamp (auto-assigned at point of creation). |
| modified | Date |  | No | Yes | Item last modified date/timestamp (autoassigned at point of update). |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Context table*** |  |  |  |  |  |
| ***Name*** | ***Type*** | ***Key*** | ***Null?*** | ***Indexed?*** | ***Value*** |
| contextkey | Integer | PK | No | Yes | Primary key (auto-assigned from a sequence). |
| itemkey | Integer | FK | No | Yes | Foreign key to item. |
| context | Varchar |  | No | Yes | Item context. |
| created | Date |  | No | Yes | Context creation date/timestamp (auto-assigned at point of creation). |

NOTE Invoking the *Delete-item* method deletes the relevant row from the *Item* table and all rows from the *Context* table related to the item via a foreign key.

## 6.3 rewrite.txt

The OPTIONAL file “rewrite.txt” is used to modify the url before fixity is performed. Each line contains a mapping used for this modification.

<*from-prefix*> <*to-prefix*> <*optional-source*>

<*from-prefix*> <*to-prefix*> <*optional-source*> …

If the <from prefix> matches the fixity url then the <to prefix> is substituted. If the optional source map is provided then that source will be used for performing the fixity operation. For example, assuming the mapping file:

http://host1:1234/storage/content/ http://host2/storage/fixity/ merritt

the URL:

http:// host1:1234/storage/content/1001/ark%3A%2F13030%2Fqt11z1k021/ 1/system%2Fmrt-ingest.txt

is automatically mapped to:

http:// host2/storage/fixity/1001/ark%3A%2F13030%2Fqt11z1k021/ 1/system%2Fmrt-ingest.txt

before the fixity test is performed. Because “merritt” is supplied as an optional third value, the fixity test is performed by the “merritt” handler and not the standard “web” handler.

# 7 Usage

## 7.1 Audit

The Audit service runs as a daemon in tomcat. To initiate processing a *resume* command is required, even at startup of the tomcat service. The speed of the fixity handling is controlled through properties in the audit-info.txt file in the Audit home directory.

|  |  |  |
| --- | --- | --- |
| ***Property Name*** | ***Format*** | ***Description*** |
| threadPool | Number > 0 | Number of threads simultaneously running fixity |
| interval | Number > -1  (in days) | This value specifies how many days need to elapse before fixity will again be allowed for a particular item. It does not guarantee that an item will be tested in that period.  0 = no elapsed period |
| queueSleep | Number > -1  (in *seconds*) | Number of seconds between the fixity service allowing the next entry to be tested. This throttle is used to spread out any clumped entry of items in the database |
| periodicReportFrequency | Number > 1  (in hours) | Number of hours between the automatic creation of a Service Status report mailed to administrator. |

## 7.2 Periodic Report

A periodic report is emailed to the administrator containing an attached Service Status report. The frequency, format and delivery address of the report are controlled through the “audit-info.txt“ file.

|  |  |
| --- | --- |
| ***Property Name*** | ***Description*** |
| periodicReportFrequency | Number of hours between the automatic creation of a Service Status report mailed to administrator. |
| periodicReportFormat | Format of the report as attached in the administrative email |
| notificationEmail | Delivery address of the email |
| supportURI | From address of the email |

To aid in the automated processing of mailed reports, the subject line of the email message MUST confirm to the Merritt template:

Subject: *service* [*instance*]: *status* -- *message*: *extra*; ...

where *service* is “Fixity”, *instance* is the service instance, “[dev]” or “[stg]” (or not provided, if production), *status* is “OK” or “Fail”; *message* is “Periodic report”; and *extra* is an optional list of report specific parameters. For example:

Subject: Fixity [dev]: OK -- Periodic report

Subject: Fixity: Fail -– Periodic report: 0 failed; 76 unavailable

## 7.3 Commands

### 7.3.1 Service Commands

The Service commands are used to control the fixity processing within the Audit service. The Response on each command is the Service State.

|  |  |
| --- | --- |
| ***Command*** | ***Function*** |
| Service state | Returns the current status of the Audit service |
| resume | Required to start fixity processing on startup  Start fixity processing  Set periodic report if not running |
| pause | Stop fixity processing |
| shutdown | Stop fixity processing  Block all commands requiring SQL access  Stop periodic report handling |

### 7.3.2 Entry commands

The Entry commands are used to build the database content used by fixity processing. The Response on each command is the Item state.

|  |  |
| --- | --- |
| ***Command*** | ***Function*** |
| Add | Run fixity on request elements  Add entry to database if fixity OK |
| Test | Run fixity on request elements |
| Queue | Add entry to database |
| Update | Update of entry  Run fixity on request elements  Add new entry to database if fixity OK |
| Delete Item | Match item on url  Delete item entry matching that URL  Delete all context entries associated with that item entry |

### 7.3.3 Requested Reports

Because of concern about database access for 10M items, all reports are delivered as email. Each command allows the user to specify the format of the report as returned as an email attachment. The response on each request is the Submission State.

|  |  |  |
| --- | --- | --- |
| ***Command*** | ***Description*** | ***Email response*** |
| Request Item Report | Return set of item states based on select criteria contained in passed context value and status types | Items report |
| SQL Report | Return SQL response based on passed select request | SQL report |

To aid in the automated processing of mailed reports, the subject line of the email message MUST confirm to the Merritt template:

Subject: *service* [*instance*]: *status* -- *message*: *extra*; ...

where *service* is “Fixity”, *instance* is the service instance, “[dev]” or “[stg]” (or not provided, if production), *status* is “OK” or “Fail”; *message* is “Periodic report”; and *extra* is an optional list of report specific parameters. For example:

Subject: Fixity [stg]: OK –- Database shutdown

Subject: Fixity: Fail – Database startup: could not restart

## 7.4 Source Modes

The source modes are used to identify a specific type of fixity handler that is required for a given url. The intent of this feature is to allow third-party users to build their own handlers for performing fixity within specialized environments. Most of the fixity service code is to support the daemon and report handling. The actual fixity processing is relatively small.

|  |  |
| --- | --- |
| ***Source*** | ***Description*** |
| Web | Pull the content to be tested from a remote service.  Perform size and digest handling based on the size and digest saved in the database |
| Merritt | Special handler for Storage Audit. The URL is a Merritt Storage fixity URL request. The handler uses the response from Storage to determine if the fixity for the component completed successfully. |

## 7.5 rewrite.txt

The “rewrite.txt” file may be optionally used to map one database URL to another. The file exists in the Audit service home directory. The mapping is a simple process of replacing some prefix in a saved URL with a different prefix before the fixity testing begins. This feature allows one database image for a repository to be used to support the fixity of another repository – primarily in the case of replication.

The file consists of one or more lines, each consisting of two or three elements delimited by one or more white space characters: SP (U+0020) or HT (U+0009. Each lined is scanned to see if the prefix for that line matches the URL being tested. If the prefix matches then the URL prefix from the database is replaced and the fixity test begins. If the prefix does not match then the next line is tested.

<*from-prefix*> WS+ <*to-prefix*> [ WS+ <*mapped-source*> ]

|  |  |  |
| --- | --- | --- |
| ***Name*** | ***Required?*** | ***Description*** |
| From prefix | yes | Prefix of database url being matched |
| To prefix | yes | Substitution prefix if “from prefix” is matched |
| Mapped source | No | Optional replacement of source (e.g. web -> merritt)  Used if “from prefix” matches |

Some things to note:

* The database URL is not modified as the result of the execution of this feature.
* Any add or test commands may also be processed for mapping.

### References

|  |  |
| --- | --- |
| [Abbott] | Daisy Abbott, *What is Digital Curation?* April 3, 2008 <http://www.dcc.ac.uk/resource/briefingpapers/what-is-digital-curation/>. |
| [Denning] | Peter J. Denning, Chris Gunderson, and Rich Hayes-Roth, “Evolutionary system development,” *Communications of the ACM* 51:17 (December 2008): 29-312010. |
| [Fielding] | Roy Fielding and Richard Taylor, “Principled design of the modern web architecture,” *ACM*  *Transactions on Internet Technology* 2:2 (May 2002): 115-150 <doi:10.1145/514183.514185>. |
| [Fisher] | David A. Fisher, *An Emergent Perspective on Interoperation in Systems of Systems*, CMU/SEI2006-TR-003, ESC-TR-2006-003, March 2006  <http://www.sei.cmu.edu/pub/documents/06.reports/pdf/06tr003.pdf>. |
| [Merritt] | UC3, *Merritt: An Emergent Approach to Digital Curation Infrastructure*, 2010. |
| [Multipart] | L. Masinter, *Returning Values from Forms: multipart/form-data*, RFC 2388, August 1989 <http://www.ietf.org/rfc/rfc2388.txt>. |
| [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>. | |