

## CDL/UC Curation Center TeakStore

Rev. 0.7 – 2009-09-03

### 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 California Digital Library (CDL) UC Curation Center (UC3) 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, 2008].

In order to meet these obligations UC3 is developing Teak, an emergent approach to digital curation infrastructure [CDL, 2009]. Teak 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, 2008]; 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 emerges from the strategic combination of individual, atomistic services [Fisher, 2006].

One of the foundational Teak services is TeakStore, which provides a robust, flexible, and easily deployed environment in which to manage the secure and persistent storage of encoded files that represent digital content.

**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 [Bradner, 1997].

### 2 TeakStore

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

- Service
- Node
- Object
- Version
- File

TeakStore provides methods that can be used to manipulate and access these entities and their state in useful ways.

## 2.1 Service

The TeakStore service itself, providing secure, persistent storage of versioned digital objects. The global service state properties **MUST** minimally include:

- Service name.
- Service identifier, assigned to be globally unique among all UC3-controlled instantiations.
- Service version.
- Customer support contact information (URI or email address).
- Enumeration of storage nodes.
- Total number of objects.
- Total number of files.
- Total number of versions.
- Total size (in octets).
- Creation date/timestamp.
- Modification date/timestamp.
- Last access date/timestamp.
- An enumeration of publicly-supported methods.
- Access URI for the service.

Additional global service state properties **MAY** be defined and managed by the service.

An instance of the TeakStore service encapsulates an arbitrary number of storage *nodes*.

## 2.2 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 TeakStore 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. The node state properties **MUST** minimally include:

- Node name.
- Node identifier, assigned to be locally unique among all nodes known by a single instance of the storage service.
- Node version.
- Total number of objects.
- Total number of files.
- Total number of versions.
- Total size (in octets).
- Creation date/timestamp.
- Modification date/timestamp.
- Last access date/timestamp.
- Underlying storage technology: magnetic-disk, magnetic-tape, optical-disk, solid-state.
- Underlying storage access modality: on-line, near-line, off-line.
- An enumeration of supported methods.
- Access URI for the node.

Additional node state properties **MAY** be defined and managed.

If the TeakStore service encompasses multiple stores, each store **MAY** be implemented in terms of different technological systems and storage hardware.

An instance of a storage node encapsulates an arbitrary number of *objects*.

### 2.3 Object

An *object* is a set of digitally encoded files whose change history is aggregated into discrete versions. The object state properties **MUST** minimally include:

- Object identifier, locally unique among all objects managed in a given storage node.
- An enumeration of versions and their creation dates.
- Total number of files.
- Total size (in octets).
- Creation date/timestamp.
- Modification date/timestamp.
- Last access date/timestamp.
- Last checksum verification date/timestamp.
- Access URI for the object.

Additional object state properties **MAY** be defined and managed by the service.

**NOTE** TeakStore has a weak definition of digital objects, especially in relation to those used by common repository services. This is by explicit design. TeakStore manages objects without *any* understanding of what information those objects represent. Higher order function implying or depending upon a conceptualization of objects as expressions of intellectual or aesthetic works *must* be provided by other systems.

An instance of a digital object encapsulates an arbitrary number of *versions*.

### 2.4 Version

A *version* is the particular configuration of an object at a point in time. The version state properties **MUST** minimally include:

- Version number.
- Total number of files.
- Total size (in octets).
- Creation date/timestamp.
- Modification date/timestamp.
- Access URI for the version.

Additional version state properties **MAY** be defined and managed by the service.

An instance of a version encapsulates an arbitrary number of *files*.

### 2.5 File

A *file* is a formatted digital manifestation of a unit of abstract content. The file state properties **MUST**

minimally include:

- File name.
- File pathname, relative to object
- File size (in octets).
- Message digest type, value, and date/timestamp of last verification. Supported digest algorithm types SHOULD minimally include:
  - Adler-32
  - CRC-32
  - MD2
  - MD5
  - SHA-1
  - SHA-256
  - SHA-384
  - SHA-512
- Creation date/timestamp.
- Modification date/timestamp.
- Access URI for the file.

Additional file state properties MAY be defined and managed by the service.

### 3 Service Interface

All Teak 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, 2002].

Containerized objects and versions MAY be requested *by value* in the following forms:

<b>Format</b>	<b>Extension</b>	<b>MIME type</b>
Tar	.tar	application/tar
Compressed tar	.tar.gz	application/x-gzip
Zip	.zip	application/zip

Containerized objects and versions, and uncontainerized files MAY be requested *by reference* in the following manifest format:

<b>Format</b>	<b>Extension</b>	<b>MIME type</b>
Checkm	.txt	text/checkm

NOTE Until such time as a formal MIME type for the Checkm format [CDL, 2009] is established at the IANA registry, the experimental MIME type “text/x-checkm” SHOULD be used.

Uncontainerized files are delivered in their original format with their original file name and extension. However, since the storage service does not manage format metadata of individual files, the *reported* format of all delivered files is “application-octet-stream”.

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

<b>Format</b>	<b>Extension</b>	<b>MIME type</b>
ANVL	.txt	text/anvl
HTML	.html	application/xhtml+xml
JSON	.json	application/json
Plain text	.txt	text/plain; charset=charset
RDF	.rdf	application/rdf+xml
XML	.xml	application/xml

NOTE Until such time as a formal MIME type for the ANVL format [CDL, 2009] is established at the IANA registry, the experimental MIME type “text/x-anvl” SHOULD be used.

### 3.1 RESTful API

The general organization of the RESTful API is to use the HTTP method to indicate the general operation, the request URI to indicate the entity that is being operated on, query string arguments to modify the method, and HTTP content negotiation headers to indicate the desired response form. The general form of the TeakStore HTTP request URI is:

```
/entity[/node[/object[/version[/file]]]]][?args...]
```

where the leading slash represents the storage service itself; *entity* indicates the entity that is the object of the request, and is one of “help”, “node”, “object”, “version”, “addVersion”, or “file”; *node* is a storage node identifier; *object* is an object identifier; *version* is a version number; and *file* is a file name. The special version number “0” always refers to the current version.

### 3.2 Command Line API

The command line API is designed to closely mimic the RESTful API.

<b>RESTful options</b>	<b>Command line options</b>		<b>Function</b>
D[= <i>level</i> ]	-D [ <i>level</i> ]	--debug [ <i>level</i> ]	Debug level
h[= <i>topic</i> ]	-h [ <i>topic</i> ]	--help [ <i>topic</i> ]	Help
<i>node</i>	-N <i>node</i>	--node <i>node</i>	Storage node identifier
	-o <i>file</i>	--output <i>file</i>	Output to file (rather than standard output)
r= <i>mode</i>	-r <i>mode</i>	--response-mode <i>mode</i>	Response mode: by-reference or by-value
R= <i>mode</i>	-R <i>mode</i>	--request-mode <i>mode</i>	Request mode: by-reference or by-value
Accept: <i>form</i>	-t <i>form</i>	--response-form <i>form</i>	Response format
Content-type:	-T <i>form</i>	--request-form <i>form</i>	Request format
v	-v	--verbose	Verbose response
V	-V	--version	Version information

## 4 Service Methods

The TeakStore service SHOULD support the following methods. Each method is first defined

abstractly and then in terms of RESTful and command shell 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 and a vertical bar “|” separates alternatives.

## 4.1 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 plain text, ANVL, XML, HTML, RDF, and JSON, with an appropriate default value.
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[?h[=method]] HTTP/1.x
UA: Host: store.cdlib.org
UA: Accept: response/form
UA:
```

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

- Command line API

```
% store -h [method] [-t form] [-o file]
% store help [-t form] [-o file]
% store getServiceState -h [-t form] [-o file]
% store getNodeState -h [-t form] [-o file]
% store getObjectState -h [-t form] [-o file]
% store getVersionState -h [-t form] [-o file]
% store getFileState -h [-t form] [-o file]
% store getObject -h [-t form] [-o file]
% store getVersion -h [-t form] [-o file]
% store getFile -h [-t form] [-o file]
% store addVersion -h [-t form] [-o file]
% store deleteObject -h [-t form] [-o file]
% store deleteVersion -h [-t form] [-o file]
```

## 4.2 Get Service State

METHOD <i>Get-service-state</i>		<i>[idempotent, safe]</i>	
—			
ResponseForm	Enum	Optional	Response form. The supported forms SHOULD include ANVL, XML, HTML, RDF, and JSON, with an appropriate default value.
RETURN	Response form	Mandatory	Global 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 /serviceState HTTP/1.x
UA: Host: store.cdlib.org
UA: Accept: response/form
UA:
```

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

- Command line API

```
% store getServiceState [-t form] [-o file]
```

### 4.3 Get Node State

METHOD <i>Get-node-state</i>			<i>[idempotent, safe]</i>
Node	Identifier	Mandatory	Node identifier
ResponseForm	Enum	Optional	Response form. The supported forms SHOULD include ANVL, XML, HTML, RDF, and JSON, with an appropriate default value.
RETURN	Response form	Mandatory	Node state.
SIDE EFFECTS	—		
ERRORS	400	Badly-formed request.	
	401	Unauthorized user agent.	
	404	Node not found.	
	415	Unsupported response form.	
	503	Service unavailable.	
	500	Service error.	

- RESTful API

```

UA: GET /nodeState/node HTTP/1.x
UA: Host: store.cdlib.org
UA: Accept: response/form
UA:

```

```

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

```

NOTE Since the default method modifier is “state” the “m=state” query string argument is unnecessary, but is defined for generality.

- Command line API

```
% store getNodeState node [-t form] [-o file]
```

#### 4.4 Get Object State

METHOD <i>Get-object-state</i>			<i>[idempotent, safe]</i>
Node	Identifier	Mandatory	Node identifier
Object	Identifier	Mandatory	Object identifier.
ResponseForm	Enum	Optional	Response form. The supported forms SHOULD include ANVL, XML, HTML, RDF, and JSON, with an appropriate default value.
RETURN	Response form	Mandatory	Object state.
SIDE EFFECTS	—		
ERRORS	400	Badly-formed request.	
	401	Unauthorized user agent.	
	404	Node not found.	
	404	Object not found.	
	415	Unsupported response form.	
	503	Service unavailable.	
	500	Service error.	

- RESTful API

```
UA: GET /objectState/node/object HTTP/1.x
UA: Host: store.cdlib.org
UA: Accept: response/form
UA:
```

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

- Command line API

```
% store getObjectState node object [-t form] [-o file]
```

## 4.5 Get Version State

METHOD <i>Get-version-state</i>			<i>[idempotent, safe]</i>
Node	Identifier	Mandatory	Node identifier.
Object	Identifier	Mandatory	Object identifier.
Version	Number	Optional	Version number, defaults to current version. The number “0” indicates current version.
ResponseForm	Enum	Deprecated	Response form. The supported forms SHOULD include ANVL, XML, HTML, RDF, or JSON, with an appropriate default value.
RETURN	Response form	Mandatory	Version state.
SIDE EFFECTS	—		
ERRORS	400	Badly-formed request.	
	401	Unauthorized user agent.	
	404	Node not found.	
	404	Object not found.	
	404	Version not found.	
	415	Unsupported response form.	
	503	Service unavailable.	
	500	Service error.	

- RESTful API

```
UA: GET /versionState/node/object/version HTTP/1.x
UA: Host: store.cdlib.org
UA: Accept: response/form
UA:
```

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

- Command line API

```
% store getVersionState node object [version] [-t form] [-o file]
```

## 4.6 Get File State

METHOD <i>Get-file-state</i>			<i>[idempotent, safe]</i>
Node	Identifier	Mandatory	Node identifier.
Object	Identifier	Mandatory	Object identifier.
Version	Number	Mandatory	Version number, defaults to current version. "0" indicates current version.
File	Identifier	Mandatory	File name.
ResponseForm	Enum	Deprecated	Response form. The supported forms SHOULD include ANVL, XML, HTML, RDF, and JSON, with an appropriate default value.
RETURN	Response form	Mandatory	File state.
SIDE EFFECTS	—		
ERRORS	400 Badly-formed request		
	405 Method not allowed		
	404 Node not found.		
	404 Object not found.		
	404 Version not found.		
	404 File not found.		
	415 Unsupported response form.		
	503 Service unavailable.		
	500 Service error.		

- RESTful API

```
UA: GET /fileState/node/object/version/file HTTP/1.x
UA: Host: store.cdlib.org
UA: Accept: response/form
UA:
```

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

- Command line API

```
% store getFileState node object version file [-t form] [-o file]
```

## 4.7 Get Object

METHOD <i>Get-object</i>			<i>[idempotent, unsafe]</i>
Node	Identifier	Mandatory	Node identifier.
Object	Identifier	Mandatory	Object identifier.
ResponseForm	Enum	Optional	Response form for by-value mode. Supported forms SHOULD include Tar and Zip, with an appropriate default value.
ResponseMode	Enum	Optional	Response mode: by-reference (default) or by-value.
RETURN	Response form	Mandatory	Object files aggregated into a single container.
	Checkm		Manifest containing network-accessible references to object files.
SIDE EFFECTS	Update last access date/timestamp.		
ERRORS	400	Badly-formed request.	
	401	Unauthorized user agent.	
	404	Node not found.	
	404	Object not found.	
	415	Unsupported response form.	
	501	Unsupported response mode.	
	503	Service unavailable.	
	500	Service error.	

- RESTful API

```
UA: GET /object/node/object?[r=mode] HTTP/1.x
UA: Host: store.cdlib.org
UA: Accept: response/form
```

```
OS: HTTP/1.x 200 OK
OS: Content-type: response/form | text/checkm
OS:
OS: container | manifest
```

- Command line API

```
% store getObject node object [-t form] [-r mode] [-o file]
```

## 4.8 Get Version

METHOD <i>Get-version</i>			<i>[idempotent, unsafe]</i>
Node	Identifier	Mandatory	Node identifier.
Object	Identifier	Mandatory	Object identifier.
Version	Number	Optional	Version number; defaults to current version. The number "0" indicates current version.
ResponseForm	Enum	Optional	Response form for by-value mode. Supported forms SHOULD include Tar and Zip, with an appropriate default value.
ResponseMode	Enum	Optional	Response mode: by-reference (default) or by-value
RETURN	Response form	Mandatory	Version files aggregated into a single container.
	Checkm		Manifest containing network-accessible references to version files.
SIDE EFFECTS	Update last access date/timestamp.		
ERRORS	400	Badly-formed request.	
	401	Unauthorized user agent.	
	404	Node not found.	
	404	Object not found.	
	404	Version not found.	
	415	Unsupported response form.	
	501	Unsupported response mode.	
	503	Service unavailable.	
	500	Service error.	

- RESTful API

```

UA: GET /version/node/object/version?[r=mode] HTTP/1.x
UA: Host: store.cdlib.org
UA: Accept: response/form
UA:
OS: HTTP/1.x 200 OK
OS: Content-type: response/form | text/ checkm
OS:
OS: container | manifest

```

- Command line API

```
% store getVersion node object [version] [-t form] [-r mode] [-o file]
```

## 4.9 Get File

METHOD <i>Get-file</i>		<i>[idempotent, unsafe]</i>	
Node	Identifier	Mandatory	Node identifier.
Object	Identifier	Mandatory	Object identifier.
Version	Number	Mandatory	Version number. The number “0” indicates current version.
File	Identifier	Mandatory	File name.
ResponseMode	Enum	Optional	Response mode: by-reference or by-value (default)
RETURN	Octet-stream	Mandatory	File content.
	Checkm		Manifest containing network-accessible reference to file content.
SIDE EFFECTS	Update last access date/timestamp.		
ERRORS	400 Badly-formed request.		
	401 Unauthorized user agent.		
	404 Node not found.		
	404 Object not found.		
	404 Version not found.		
	404 File not found.		
	415 Unsupported response mode.		
	503 Service unavailable.		
500 Service error.			

- RESTful API

```

UA: GET /file/node/object[/version]/file?[r=mode] HTTP/1.x
UA: Host: store.cdlib.org
UA:
OS: HTTP/1.x 200 OK
OS: Content-type: mime/type | text/checkm
OS:
OS: content | manifest
    
```

- Command line API

```
% store getFile node object version file [-t form] [-r mode] [-o file]
```

### 4.10 Add Version

METHOD <i>Add-version</i>			[ <i>non-idempotent, unsafe</i> ]
Node	Identifier	Mandatory	Node identifier.
Object	Identifier	Mandatory	Object identifier.
RequestMode	Enum	Optional	Request mode: by-reference (default) or by-value.
URI	URI	Mandatory	If request mode is by-reference, URI of manifest containing network accessible references to version files.
Manifest	Checkm		If request mode is by-reference, manifest containing network-accessible references to version files.
Size	Number	Optional	Manifest size, in octets.
MessageDigest Type	Enum	Optional	Manifest message digest type. The supported types SHOULD include Adler-32, CRC-32, MD2, MD5, SHA-1, SHA-256, SHA-384, or SHA-512.
MessageDigest Value	String	Optional	Hexadecimal representation of the manifest message digest value.
ResponseForm	Enum	Optional	Response form. The supported forms SHOULD include ANVL, XML, HTML, RDF, or JSON, with an appropriate default value.
RETURN	Response form	Mandatory	Newly created version state.
SIDE EFFECTS	<p>If the object does not exist, create an empty object; increment current version number; populate the version; establish appropriate version state; and update object, node, and service state.</p> <p>Note that <i>any</i> change introduced to an object SHALL result in a new version. In particular, the concept of a <i>replace</i> or <i>update-in-place</i> method is <i>not</i> supported by TeakStore.</p>		
ERRORS	400 Badly-formed request.		
	401 Unauthorized user agent.		
	404 Node not found.		
	404 Object not found.		
	400 Unsupported request mode.		
	415 Unsupported request form.		
	415 Unsupported response form.		
	413 Version too large.		
	400 Empty version		
	400 Duplicate version		
	503 Service unavailable.		
500 Service error.			

- RESTful API

```

UA: POST /addVersion/node/object HTTP/1.x
UA: Host: store.cdlib.org
UA: Accept: response/form
UA: Content-type: application/x-www-form-urlencoded | text/checkm
UA: Content-length: size
UA:
UA: manifest-uri=uri[&manifest-size=size][&digest-type=type&
  digest-value=value] | manifest
  
```

```
OS: HTTP/1.x 201 CREATED
OS: Content-type: response/form
OS: Location: http://store.cdlib.org/version/node/object/version
OS:
OS: state
```

- Command line API

```
% store addVersion node object manifest [-t form] [-o file]
```

#### 4.11 Delete Object

METHOD <i>Delete-object</i>		<i>[idempotent, unsafe]</i>	
Node	Identifier	Mandatory	Node identifier.
Object	Identifier	Mandatory	Object identifier.
ResponseForm	Enum	Optional	Response form. The supported forms SHOULD include ANVL, XML, HTML, RDF, or JSON, with an appropriate default value.
RETURN	Response form	Mandatory	Deleted object state.
SIDE EFFECTS	Delete the object and update node and service state.		
ERRORS	400 Badly-formed request.		
	401 Unauthorized user agent.		
	404 Node not found.		
	404 Object not found.		
	415 Unsupported response form.		
	503 Service unavailable.		
	500 Service error.		

- RESTful API

```
UA: DELETE /object/node/object HTTP/1.x
UA: Host: store.cdlib.org
UA: Accept: response/form
UA:
```

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

- Command line API

```
% store deleteObject node object [-t form] [-o file]
```

### 4.12 Delete Version

METHOD <i>Delete-version</i>			<i>[idempotent, unsafe]</i>
Node	Identifier	Mandatory	Node identifier.
Object	Identifier	Mandatory	Object identifier.
Version	Number	Mandatory	Version number. The number “0” indicates the current version.
ResponseForm	Enum	Optional	Response form. The supported forms SHOULD include ANVL, XML, HTML, RDF, or JSON, with an appropriate default value.
RETURN	Response form	Mandatory	Deleted version state.
SIDE EFFECTS	Delete the object version and update object, node, and service state.		
ERRORS	400	Badly-formed request.	
	401	Unauthorized user agent.	
	404	Node not found.	
	404	Object not found.	
	404	Version not found.	
	415	Unsupported response form.	
	503	Service not available.	
	500	Service error.	

- RESTful API

```

UA: DELETE /version/node/object/version HTTP/1.x
UA: Host: store.cdlib.org
UA: Accept: response/form
UA:

```

```

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

```

- Command line API

```
% store deleteVersion node object version [-t form] [-o file]
```

## 5 Implementation Notes

TeakStore exposes its function via three interactive modalities:

- A Java J2SE 1.6 API
- A command line application based on the Java API
- A web interface based on Jetty, Jersey, and the Java API

Although the method names (e.g. “getServiceState”, “addVersion”) used in the TeakStore command line application are documented above in camel-case, they **SHALL** be parsed and matched in a case-insensitive manner. Similarly, although the documented method syntax uses the terse, single letter form of command options (e.g. “-t *form*”), the analogous keyword variants (“--response-format *form*”) **MUST** also be accepted. The command line application **SHALL** support the “-v” and “--version” (version) command options.

Although a creation date/timestamp is documented above as part of the service, node, object, version, and file state, it **SHALL** not be managed or reported.

NOTE File and directory creation times are not directly retrievable in J2SE 1.6.

TeakStore uses CAN (Content Access Node) as the implementation for its storage nodes [CDL, 2009]; Paritree to structure the branches of the CAN hierarchical object stores [CDL, 2009]; Dflat to structure the leaves of the hierarchical object stores [CDL, 2009]; Checkm to define Dflat manifests [CDL, 2009]; and ReDD to delta-compress Dflat object versions [CDL, 2009].

TeakStore imposes the following additional requirements beyond these specified by the CAN specification:

- The Namaste tag “0=CAN\_0.4” is **REQUIRED**.
- The global properties file “can-info.txt” is **REQUIRED** and **SHALL** express the “Name”, “Node-scheme”, “Branch-scheme”, and “Leaf-scheme” properties.
- The log directory “log/” is **REQUIRED** and **SHALL** hold the file “last-access.txt” containing the date/timestamp of the last end user access of an object managed in the CAN.

TeakStore imposes the following additional requirements beyond these specified by the Dflat specification:

- The Namaste tag “0=Dflat\_0.15” is **REQUIRED**.
- The current version file “current.txt” is **REQUIRED**.
- The global properties file “dflat-info.txt” is **REQUIRED** and **SHALL** express the “Object-scheme”, “Manifest-scheme”, “Full-scheme”, “Delta-scheme”, and “Current-scheme” properties.
- The log directory “log/” is **REQUIRED** and **SHALL** hold the files “last-access.txt” and “last-fixity.txt” containing the date/timestamps of the last end user access and last fixity check of the object, respectively.
- The current version sub-directory **SHALL** contain a version manifest file “manifest.txt”.
- All non-current version sub-directories **SHALL** be delta-compressed representations and **SHALL** contain delta directory and version manifest files, “d-manifest.txt” and “manifest.txt”, respectively.

A TeakStore is instantiated in a file system with the following structure:

```
<store_home>/
    0=teakstore_0.5
    admin/
    [ lock.txt ]
    log/
    nodes.txt
    teakstore-info.txt
```

The file “0=teakstore\_0.5” is the service’s Namaste signature.

Various TeakStore directories MAY contain a file named “lock.txt”, which indicates that the directory (and all dependent sub-directories) are being processed in a manner that may put it in an unknown, inconsistent, or incomplete state. The file holds the date/timestamp, in fully-qualified W3C form [DateTime], at which the lock was established and an identifier of the process holding the lock (such as a process or thread identifier), for example:

```
Lock: 2009-02-14T11:04:23+0800 50124
```

Any process intending to manipulate a TeakStore directory in a manner that affects its state MUST first obtain a write lock on that directory. Any process holding a write lock on a directory SHOULD release it at the earliest safe moment. All processes reading a directory SHOULD look for the presence of the lock file before attempting the operation; if present, the process SHOULD continue only under an assumption that the data read may be incomplete or inconsistent.

The file “teakstore-info.txt” declares the global TeakStore properties, for example:

```
Name: store.cdlib.org
Service-scheme: TeakStore/0.5
Node-scheme: CAN/0.6
Verify-on-read: true
Verify-on-write: true
Access-uri: http://store.cdlib.org/
Support-uri: mailto:teak-support@ucop.edu
```

The file “nodes.txt” defines the nodes known to the TeakStore. Nodes are defined by name and access URI, for example:

```
can01 http://can01.cdlib.org/
can02 http://can02.cdlib.org/
can03 file:home/can03
...
```

The administrative directory “admin/” holds administrative declarations associated with the TeakStore itself, as opposed to the objects managed in it.

```
admin/
    [ lock.txt ]
    summary-stats.txt
```

The administrative directory contains a file named “summary-stats.txt” that holds summary statistics about the TeakStore expressed as ANVL name/value pairs, for example:

```
Node-count: 3
Object-count: 18302
Version-count: 27551
File-count: 405833
Total-size: 730415172
```

The “Node-count”, “Object-count”, “Version-count”, and “File-count” properties indicate the number of nodes, objects, versions, and files, respectively, managed in the TeakStore.

The “Total-size” property indicates the total size of all versioned files, in octets.

The logging directory “log/” holds log information about the use of the TeakStore and the objects managed in it.

```
log/
  cmd-access.txt
  [ diagnostics.txt ]
  [ last-access.txt ]
  [ last-fixity.txt ]
  [ lock.txt ]
  web-access.txt
```

The directory SHALL contain a file named “cmd-access.txt” containing a log entry for each invocation of a Storage service method via the command line API. The directory SHALL contain a file named “web-access.txt” containing a log entry for each invocation of a Storage service method via the web API. The format of these log files is the Combined Log Format (CLF).

```
ip - - date-time "request-uri" status size "referer" "user-agent"
```

where “ip” is the IP address of the client; “date-time” is the date/timestamp of the request; “request-uri” is the request URI; “status” is the status of the response; “size” is the size of the response, in octets; “referer” is the referring URI; and “user-agent” is the requesting user agent. For the command line log, “ip” SHOULD be set to the TeakStore name (the “Name” property in the “teakstore-info.txt” file); “request-uri” SHOULD be set to the equivalent web API syntax; “referer” SHOULD be set to the current working directory (e.g. Java `user.dir` or Unix `$pwd` property); and “user-agent” SHOULD be set to the command shell login name of the user (e.g..Java `user.name` or Unix `$user` property)

The directory MAY contain a file named “diagnostics.txt” containing diagnostic logging information.

The directory MAY contain a file named “last-access.txt” containing the date/timestamp of the last end-user access of an object in the TeakStore in fully-qualified W3C form and an identifier of the accessing process, for example:

```
Last-access: 2009-01-29T05:33:24+0800 66212
```

The directory SHOULD contain a file named “last-fixity.txt” containing the date/timestamp in fully-qualified W3C form of the last complete fixity verification of the objects in the TeakStore and an identifier of the fixity process, for example:

Last-fixity: 2008-12-22T23:00:10+0800 18002

## References

- Abbott, Daisy, *What is Digital Curation?* April 3, 2008 <<http://www.dcc.ac.uk/resource/briefing-papers/what-is-digital-curation/>>.
- Bradner, S., *Key Words for Use in RFCs to Indicate Requirement Levels*, BCP 14, RFC 2119, March 1997 <<http://www.ietf.org/rfc/rfc2119.txt>>.
- California Digital Library, *ANVL: A Name Value Language*, 2009.
- California Digital Library, *CAN: A Simple File System-Based Object Store*, 2009.
- California Digital Library, *Checkm: A Checksum-Based Manifest Format*, 2009.
- California Digital Library, *Dflat: A Simple File System Convention for Digital Object Storage*, 2009.
- California Digital Library, *Digital Preservation Program: Foundations*, 2009.
- California Digital Library, *Reverse Directory Deltas (ReDD)*, 2009.
- California Digital Library, *Teak: An Emergent Approach to Digital Curation Infrastructure*, 2009.
- Denning, Peter J., Chris Gunderson, and Rich Hayes-Roth, "Evolutionary system development," *Communications of the ACM* 51:17 (December 2008): 29-31.
- Fielding, Roy, 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>.