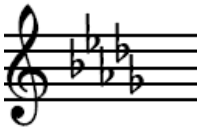

Dflat: A Simple File System Convention for Object Storage

Rev. 0.11 – 2009-04-07

1 Introduction

Dflat is a simple file system convention for storing a digital object. It imposes minimal architectural and policy constraints while reserving a small set of file system names (directories and files) that place certain salient object features, if present, in well-known locations within a single directory hierarchy that houses the entire object. With no other knowledge about the nature or purpose of the object, human administrators and automated agents that visit this "digital flat" will be able to identify its "occupant" from a small text file of derived metadata (a kind of nameplate) and identify "rooms", some or all of which may have been built out, containing such things as content, metadata, version history, annotations, and administrative records. Dflat provides the sort of shallow knowledge sufficient to isolate various object components that make it easy to build a range of common object repository functions, from basic trouble-shooting to fixity checking and replication.

Intentionally excluded from this specification are deeper assumptions about how object collections are arranged, under what policy regimes objects are managed, and how objects are used. If more complex functions are desired, it is expected that implementers will supplement Dflat incrementally by addition of deeper assumptions, requirements, and recommendations from other specifications. While Dflat can be deployed usefully on its own, it was designed to interoperate cleanly with other independent, but related specifications such as CAN (Content Access Node) [CAN] and Pairtree [Pairtree]. All three of these specifications start from the assumption that a file system is a sufficient storage abstraction for the effective management of digital objects. Assuming that these objects are stored in a tree-like directory hierarchy, CAN specifies the organization and global properties of the tree; Pairtree specifies the form of the branches of the tree; and Dflat specifies the local structure of the leaves of the tree.

2 Notation

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

Angle brackets and italics are used to indicate an arbitrary, as opposed to prescribed, file or directory name; for example, the arbitrarily-named payload file "*<payload>*". When referring to directory names or symbolic links in discursive text or examples, the names are followed by a solidus ("/") or commercial at sign ("@"), respectively, for example "`directory`/" or "`symlink`@"; these suffixes are indicative of type and are not part of the names. All directory and file names are case sensitive.

All uses of the term "directory" in this document MAY be interchanged with "folder" without loss of intended meaning. Similarly, all uses of the solidus ("/") as a directory path separator MAY be interchanged with a reverse solidus ("\"). In other words, this specification SHOULD be applicable in both Unix/Linux and Windows/DOS contexts.

In examples of file system directory hierarchies, non-required directories or files are enclosed by square brackets ("[" and "]"), a number sign ("#") introduces an informative comment, and an ellipsis ("`...`") indicates arbitrary repetition of the previous element.

Augmented Backus-Naur Form (ABNF) [RFC5234] is used to define the syntax of specific files required or recommended by this specification. Syntax rules names left undefined in this specification (for example, *ALPHA*) SHALL be interpreted as core ABNF names.

The complete set of Dflat conventions is provided in Appendix A.

3 A Digital Flat

A Dflat, or "digital flat", is a file system hierarchy that completely contains a single digital object and only that object. The directory that is the structural root of a Dflat file system hierarchy is known as the Dflat *home directory*. Only the files and directories defined in this and subsequent sections MAY exist in and beneath a Dflat home directory. The Dflat specification reserves a few key file system names within the home directory and its sub-directories, but it does not dictate how the home directory itself is named, as that name is not visible from inside the Dflat; nonetheless, selection of a home directory naming convention facilitates collection management and MAY be subject to other specifications (such as CAN).

A Dflat looks like this:

```
<dflat_home>/                   # Dflat home directory
             current@           # symlink to current version
             dflat-info.txt     # Dflat signature file
           [ lock.txt ]         # write lock
           [ log/ ]             # log directory
           [ v001/ ]            # object version directories
           [ v002/ ]
           [ ...   ]
```

A Dflat home directory MUST contain a file named "`dflat-info.txt`" that functions as the indicative Dflat signature and defines the global properties of the Dflat itself.

The home directory MAY contain a file named "`lock.txt`" that indicates an in-process write operation on the Dflat that puts it into a temporarily-inconsistent state.

The home directory MAY contain a sub-directory named "`log`/" to hold log information about the use of the object.

The home directory SHALL contain a sub-directory named "`v001`/" holding the initial version of the object. The home directory MAY contain one or more additional sub-directories of the form "`v`*<nnn>*/" holding subsequent versions of the object. The home directory SHALL contain a symbolic link named '`current@`" that points to the current object version sub-directory.

3.1 Dflat Signature File (dflat-info.txt)

The REQUIRED "`dflat-info.txt`" file self-identifies a directory as a Dflat and defines the properties of that Dflat itself, as opposed to the object stored in it. These properties are specified in terms of ANVL [ANVL] name/value pairs; for example:

```
  This: Dflat/0.10
[ Manifest-scheme: Checkm/0.1 ]
[ Delta-scheme: ReDD/0.1 ]
[ Class-scheme: CLOP/0.3 ]
```

The signature file MUST define the "`This`" property, whose value MUST indicate the version of the Dflat specification to which the Dflat conforms.

If the Dflat includes object version file manifests then the signature file MUST define the "`Manifest-scheme`" property to indicate the type of manifest; otherwise this property is OPTIONAL.

If a delta scheme is used by the Dflat to reduce redundant storage of object versions then the signature file MUST define the property "`Delta-scheme`" to indicate the mechanism used to manage changes between versions; otherwise this property is OPTIONAL.

The OPTIONAL "`Class-scheme`" property MAY be used to specify the organizational convention used for managing the administrative properties of managed objects.

3.2 Lock file (lock.txt)

The "`lock.txt`" file is OPTIONAL. If present it indicates that an administrative user is processing the Dflat in a manner that may temporarily put it in an unknown or inconsistent state, in other words, performing a write rather than read operation. The file holds the fully-qualified date/timestamp, in W3C form [DateTime], at which the lock was established and the name of the user agent holding the lock.

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

All processes reading the Dflat SHOULD look for the presence of the lock file before attempting the operation. All processes establishing a write lock on a Dflat SHOULD release the lock at the earliest safe moment.

3.3 Log Directory (log/)

The OPTIONAL "`log`/" directory contains log information about the use of the object

```
log/
  [ last-access.txt ]
  [ last-fixity.txt ]
```

The "log/" directory SHOULD contain a file named "`last-access.txt`" holding the fully-qualified date/timestamp of the last end-user access of the object in W3C form.

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

The directory SHOULD contain a file named "`last-fixity.txt`" holding the fully-qualified date/timestamp in W3C form of the last fixity verification of the object.

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

3.4 Version Directories (current@, v<*nnn*>/)

In order to manage changes introduced to an object over time the Dflat home directory SHALL contain one or more sub-directories of the form "v<nnn>/", each holding information sufficient to re-instantiate the state of the object at an arbitrary point in time. The symbolic link "`current@`" SHALL point to the sub-directory holding the current object version, for example:

```
<dflat_home>/
            current@              # symlink to version 2 -> ./v002
            dlfat-info.txt
            v001/                 # version 1
            v002/                 # current version 2
```

Object versions are numbered sequentially, starting with 1 and incrementing by 1. These version numbers are reflected in the names of the version sub-directories, "`v001/`", "`v002/`", "`v003/`", … For version numbers less than 100 the version sub-directory names are zero padded to 3 digits. For version numbers equal to or greater than 100 the version sub-directory names are not padded, thus the full sequence will look like:

```
v001/, v002/, ..., v998/, v999/, v1000/, v1001/, ..., /v12345, ...
```

The complete object state represented by a version sub-directory is:

```
[ admin/ ]                        # administrative declarations
[ annotation/ ]                   # data from object users
[ data/ ]                         # data form object producer
[ enrichment/ ]                   # data from automated processes
[ lock.txt ]                      # write lock
[ manifest.txt ]                  # file manifest
[ relationships.ttl ]             # inter-file relationships
[ splash.txt ]                    # summary metadata
```

The OPTIONAL "`admin/`" sub-directory is reserved to hold administrative declarations about the object.

The OPTIONAL "`annotations`/" sub-directory is reserved to hold user-supplied payload files, which MAY contain either object data or metadata. The OPTIONAL "`data`/" sub-directory is reserved to hold object producer- or curator-supplied payload files. The OPTIONAL "`enrichment/`" sub-directory is reserved to hold system-derived payload files. Dflat reserves no names, and imposes no constraints on structure, within the "`annotations`/", "`data/`", or "`enrichment`/" directories, so the object producer (or curator) has complete freedom to define files in this space. To facilitate interoperation it is RECOMMENDED that no data payload files have a filename length greater than 255 characters.

The OPTIONAL "`lock.txt`" file is reserved to indicate that a write operation is in process on the version directory that puts it into a temporarily-inconsistent state. This file shares the same syntax as

the home directory-level "`lock.txt`" file.

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

The RECOMMENDED "manifest.txt" file is reserved to provide a manifest of all files (besides itself) that comprise an object version.

The OPTIONAL "`relationships.ttl`" file is reserved to define typed inter-file relationships in Turtle form [Turtle].

The RECOMMENDED "`splash.txt`" file is reserved to provide summary Dublin Core Kernel metadata [Kernel].

3.4.1 Admin Directory (admin/)

The OPTIONAL "`admin`/" directory holds administrative declarations for the object. These MAY take the form of CLOP class memberships [CLOP].

```
admin/
    [ <facet>-class.txt ]
```

3.4.2 Payload Directories (annotations/, data/, enrichment/)

The OPTIONAL payload directories are reserved to hold object payload files provided by the object's users, producers or curators, and automated system processes, respectively. Payload files MAY be nested in arbitrary sub-directory hierarchies and both files and directories MAY have arbitrary names, for example:

```
data/
     <payload-file>
     ...
     <sub-directory>/
     ...
```

3.4.3 Manifest File (manifest.txt)

The RECOMMENDED "`manifest.txt`" file is reserved to hold a Checkm [Checkm] manifest of all files that comprise an object version.

3.4.4 Relationships File (relationships.txt)

The OPTIONAL "`relationships.ttl`" file is reserved to hold inter-file relationships defined in terms of Turtle [Turtle] triples.

```
<source> <relationship> <target>
```

The source and target component of a triple MAY be a file pathname, a file pattern resolvable to a set of file pathnames or non-file entities.

A file pattern is a regular expression using Bourne shell metacharacters in which "?" indicates a single

arbitrary character, and "*" indicates an arbitrary length character string. The special non-file patterns "`.`"and "`..`" refer to the entire object version and the entire object, respectively.

The allowable relationships are drawn from those defined by Fedora [Fedora], OAI-ORE [ORE] and other similar ontologies and vocabularies, and minimally include:

```
is-annotation-of
is-dependent-on
is-derivative-of
is-description-of
is-newer-version-of
is-part-of
```

3.4.5 Splash Metadata File (splash.txt)

The RECOMMENDED "`splash.txt`" file is reserved to hold Dublin Core Kernel metadata [Kernel] for the object expressed as ANVL name/value pairs; for example:

```
who: Twain, Mark
what: Pudd'nhead Wilson
when: 1894
where: 130.13/1838300383848
```

This metadata is intended to provide a quick summary of the object; more extensive documentation of an object's intellectual, administrative, technical, and structural properties MAY be placed in the object's "`data/`", "`enrichment/`", or "`annotations/`" sub-directories.

4 Implementation

The Dflat specification was developed with the intention that it could be implemented on top of any file system that supports hierarchical directory structures, arbitrary directory and file names, and symbolic links to directories, such as POSIX [POSIX].

5 Security Considerations

Dflat poses no direct risk to computers or networks. As a file system convention, Dflat is capable of holding files that might contain malicious executable content, but it is no more vulnerable in this regard than any file system.

Appendix A: Complete Dflat Conventions

The overall file system structure of a Dflat is:

```
<dflat_home>/
           [ current@ ]
             dflat-info.txt
           [ lock.txt ]
           [ log/ ]
           [ v<nnn>/ ]
             ...
```

The generic production rules for all ANVL-based files are:

```
<file>      = 1*<line>
<line>      = <name> ":" 1*WSP <value> EOL
<name>      = 1*VCHAR
<value>     = 1*VCHAR
EOL         = CR / CRLF / LF
```

More specific rules MAY be defined by each such ANVL-based file. It is RECOMMENDED that only a single white space character (WSP) is used to separate the name/value pair.

The production rules for the Dflat signature file "`dflat-info.txt`" are:

```
<signature> = <this> EOL 0*<property>
<this>      = "This:" 1*WSP "Dflat/" <version>
<property>  = (<manifest> / <delta> / <class>) 1*WSP <value> EOL
<manfiest>  = "Manifest-scheme:"
<delta>     = "Delta-scheme:"
<class>     = "Class-scheme:"
<version>   = <major> "." <minor>
<major>     = NONNEG
<minor>     = NONNEG
NONNEG      = "0" / (1*POSDIG 0*DIGIT)
POSDIG      = "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9"
```

The production rules for the manifest file "`manifest.txt`" are:

```
<manifest>  = 1*<record>
<record>    = <size> 1*WSP <type> 1*WSP <value> 1*WSP <filename> EOL
<size>      = <nonneg>
<type>      = "Adler-32" / "CRC-32" / "MD5" / "SHA-1" / "SHA-256" /
              "SHA-384"  / "SHA-256"
<value>     = 1*(HEXDIG / "a" / "b" / "c" / "d" / "e" / "f")
```

It is RECOMMEDED that only a single white space character separates the record items. It is RECOMMENDED that all message digest values use either upper- or lower-case hexadecimal characters.

The production rule for the file "last-access.txt" is:

```
<last-access> = "Last-access:" 1*WSP <date-time>
<date-time>   = <date> "T" <hr> ":" <mn> ":" <se> "+" <zzzz>
<date>        =  <year> "-" <mo> "-" <dy>
```

The production rule for the file "last-fixity.txt" is:

```
<last-access> = "Last-access:" 1*WSP <date-time>
```

The production rule for the file "`lock.txt`" is:

```
<lock>        = "Lock:" 1*WSP <date-time> 1*WSP <agent>
<agent>       = 1*VCHAR
```

The production rule for the "<*facet*>-class.txt" membership files is:

```
<membership>  = 1*<class>
<class>       = 1*VCHAR EOL
```

References

[ANVL] J. Kunze, B. Kahle, J. Masanes, and G. Mohr, *A Name-Value Language (ANVL)*, Internet draft, February 14, 2005 <http://www.ietf.org/internet-draft/draft-kunze-anvl-01.txt>.

[Checkm] J. Kunze, *Checkm: A Checksum-Based Manifest Format*, Februrary 11, 2009.

[CSDGM] Federal Geographic Data Committee, *Content Standard for Digital Geospatial Metadata*, FGDC-STD-001-1998, June 1998 <http://www.fgdc.gov/standards/projects/FGDC-standards-projects/metadata/base-metadata/v2_0698.pdf>.

[DateTime] Misha Wolf and Charles Wicksteed, *Date and Time Formats*, September 15, 1997 <http://www.w3.org/TR/NOTE-datetime>.

[DCRDF] Mikael Nilsson, Andy Powell, Pete Johnson, and Admjörn Naeve, *Expressing Dublin Core metadata using the Resource Description Framework (RDF)*, DCMI Recommendation, January 14, 2008 <http://dublincore.org/documents/2008/01/14/dc-rdf/>.

[DCXML] Andy Powell and Pete Johnson, *Guidelines for implementing Dublin Core in XML*, DCMI Recommendation, April 2, 2004 <http://dublincore.org/documents/2003/04/02/dc-xml-guidelines/>.

[ERC] J. Kunze and A. Turner, *Kernel Metadata/ERC Application Profile*, Draft v1.4a, February 20, 2008, <http://dublincore.org/kernelwiki/FrontPage?action=AttachFile&do=get&target=KernelMetadataERCApplicationProfiles1_4a.htm>.

[Fedora] Fedora Commons, *Digital Object Relationships*, August 14, 2008 <http://www.fedora-commons.org/confluence/display/FCR30/Digital+Object+Relationships>.

[FIPS1802] FIPS PUB 180-2, *Secure Hash Standard*, August 1, 2002 <http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf>.

[IEEE8023] IEEE Std 802.3-2005, *IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements, Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*.

[Kernel] J. Kunze and A. Turner, *Kernel Metadata and Electronic Resource Citations (ERCs)*, v1.1, October 10, 2007 <http://dublincore.org/kernelwiki/FrontPage?action=AttachFile&do=get&target=Kernel1_1.html>.

[MARCXML] Library of Congress , *MARC 21 XML Schema*, July 3, 2008 <http://www.loc.gov/standards/marcxml/>.

[METS] Library of Congress, *METS Schema, & Documentation*, September 15, 2008 <http://www.loc.gov/standards/mets/mets-schemadocs.html>.

[MODS] Library of Congress, *Metadata Object Description Schema: MODS Schemas*, January 24, 2008 <http://www.loc.gov/standards/mods/mods-schemas.html>.

[ORE] Carl Lagoze, Herbert Van de Sompel, Pete Johnson, Michael Nelson, Robert Sanderson, and Simeon Warner, eds., *ORE Specification – Vocabulary*, October 17, 2008 <http://www.openarchives.org/ore/1.0/vocabulary#relationships >.

[Pairtree] J. Kunze, M. Haye, E. Hetzner, M. Reyes, and C. Snavely, *Pairtrees for Object Storage (V0.1)*, Internet draft, November 25, 2008 <http://www.ietf.org/internet-drafts/draft-kunze-pairtree-01.txt>.

[Par2] Michael Nahas, Peter Clements, Paul Nettle, and Ryan Gallagher, *Parity Volume Set Specification 2.0*, May 11, 2003 <http://parchive.sourceforge.net/docs/specifications/parity-volume-spec/article-spec.html>.

[POSIX] ISO/IEC 9945:2003/IEEE Std 1003.1, *Information technology – Portable operating system interface (POSIX) – Part 2: System interface*.

[ReDD] J. Kunze, *Reverse Directory Deltas (ReDD)*, draft, March 17, 2009.

[RFC1321] R. Rivest, *The MD5 Message-Digest Algorithm*, RFC 1321, April 1992 <http://www.ietf.org/rfc/rfc1321.txt>.

[RFC1950] P. Deutsch and J-L. Gailly, *ZLIB Compressed Data Format Specifications version 3.3*, RFC 1950, May 1996 <http://www.ietf.org/rfc/rfc1950.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>.

[RFC5234 D. Crocker (ed.) and P. Overell, *Augmented BNF for Syntax Specifications: ABNF*, STD 68, RFC 5234, January 2008 <http://www.ietf.org/rfc/rfc5234.txt>.

[Turtle] Dave Beckett, *Turtle – Terse RDF Triple Language*, November 20, 2007 < http://www.dajobe.org/2004/01/turtle>.