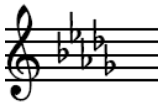
UC3 Teak 

CDL/UC Curation Center

Dflat: A Simple File System Convention for Digital Object Storage

Rev. 0.16 – 2009-08-31

1 Introduction

Dflat, and its subsidiary specification, Dnatural, together define 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 (for directories and files) that place certain salient object features, if available, 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 a "digital flat" will be able to identify its "occupant" and understand certain of its properties such as content, metadata, version history, annotations, and administrative records. Dflat and Dnatural provide the sort of shallow knowledge sufficient to isolate various object components in order to make it easy to build a range of common object repository functions, from basic trouble-shooting to fixity checking and replication. Generally speaking, Dflat defines an organizational structure for managing the versioned change of object state over time while Dnatural defines an organizational structure for representing object state at a point in time.

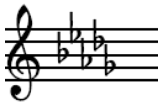
Intentionally excluded from this specification are deeper assumptions about how object collections are arranged, under what policy regimes objects are managed, and how objects are used. If more complex functions are desired, it is expected that implementers will supplement Dflat and Dnatural incrementally by addition of deeper assumptions, requirements, and recommendations from other specifications. While Dflat and Dnatural can be deployed usefully on their own, they are designed to interoperate cleanly with other independent, but related, specifications such as CAN (Content Access Node) [CAN], Pairtree [Pairtree], Checkm [Checkm], and ReDD [ReDD]. All of these specifications are based on the assumption that a file system is an appropriate storage abstraction for the effective management of digital objects. Assuming that these objects are arranged 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" 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 suffix characters are indicative of type and are not part of the names. All directory and file names are case sensitive.

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 file content required or recommended by this specification. Syntax rule names left undefined in this specification (for example, ALPHA) SHALL be interpreted as core ABNF names.

This specification is intended to be applicable, and implementable, in both Unix/Linux and Windows/DOS environments. Consequently, all uses of the term "directory" are interchangeable with "folder" without loss of meaning. Similarly, all uses of the solidus ("/") as a directory path separator are interchangeable with a reverse solidus ("\").

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

3 A Digital Flat

A Dflat, or "digital flat", is a file system hierarchy that completely encapsulates a single digital object. The directory that is the structural root of a Dflat file system hierarchy is known as the Dflat *home directory*. The Dflat specification reserves a few key file system names within the home directory and its descendent sub-directories, but it does not dictate how the home directory itself is named, as that name is not visible from inside the Dflat itself. Nevertheless, selection of a home directory naming convention facilitates collection management and MAY be subject to other specifications (such as Pairtree).

A Dflat looks like this:

```
<dflat_home>/                       # Dflat home declarations
           [ 0=dflat_<version> ]    # Dflat Namaste signature
           [ admin/ ]               # administrative directory
           [ current.txt ]          # pointer to current version
           [ dflat-info.txt ]       # Dflat properties file
           [ lock.txt ]             # write lock
           [ log/ ]                 # log directory
             v001/                  # object version directories
           [ v002/
             ...  ]
```

A Dflat home directory MAY contain a file named "0=dflat_<*version*>" that is its Namaste [Namaste] signature. A Namaste signature plays the same role for a directory that a magic number plays for a file.

The home directory MAY contain a sub-directory named "admin/" that holds non-versioned administrative declarations about the object stored in the Dflat.

The home directory SHOULD contain a file named "current.txt" that holds the name of the current object version sub-directory.

The home directory SHOULD contain a file named "dflat-info.txt" that declares the non-versioned global properties of the Dflat itself, as opposed to the properties of the object stored in the Dflat.

The directory MAY contain a file named "lock.txt" that indicates a write operation is in-process and that the Dflat may be in a temporarily unknown, inconsistent, or incomplete state.

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

The home directory MUST 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.

Note that none of the files directly found in the home directory are versioned. Versioned information is only found version sub-directories.

3.1 Namaste signature file (0=dflat_*<version>*)

The RECOMMENDED Namaste signature file "`0=dflat_`*<version>*" self-identifies a directory as a Dflat home directory. The "*<version>*" portion of the file name asserts the version of the Dflat specification to which the directory conforms. The contents of the file MUST exactly duplicate the file name.

```
0=dflat_<version>
```

Note that this value is duplicative of the "`Object-scheme`" property of the global properties file "`dflat-info.txt`". If both are present, the global properties file is considered authoritative.

3.2 Administrative directory (admin/)

The OPTIONAL administrative directory "`admin`/" holds non-versioned administrative declarations associated with the object stored in the Dflat.

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

The administrative directory MAY contain a file named "`lock.txt`" that conforms to the syntax, semantics, and normative obligations defined in Section § **3.5**.

The administrative directory MAY contain a file named "`summary-stats.txt`" holding summary statistics about the Dflat expressed as ANVL name/value pairs, for example:

```
[ Version-count: 3 ]
[ File-count: 2405 ]
[ Total-size: 3041572 ]
```

The "`Version-count`" and "`File-count`" properties indicate the number of versions and files, respectively, managed in the Dflat.

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

3.3 Current version file (current.txt)

The REQUIRED current version file "`current.txt`" holds the name of the current version sub-directory.

v<*nnn*>

3.4 Properties file (dflat-info.txt)

The RECOMMENDED properties file "`dflat-info.txt`" defines the non-versioned global properties of that Dflat itself, as opposed to the object stored in it. These properties are expressed in terms of ANVL [ANVL] name/value pairs, for example:

```
[ Object-scheme: Dflat/0.15 ]
[ Manifest-scheme: Checkm/0.1 ]
[ Full-scheme: Dnatural/0.12 ]
[ Delta-scheme: ReDD/0.1 ]
[ Current-scheme: file ]
[ Class-scheme: CLOP/0.3 ]
```

The "`Object-scheme`" property indicates the version of the Dflat specification to which the Dflat conforms. Note that this value is duplicative to the information provided by the OPTIONAL Namaste tag. If both are present, the properties file is considered authoritative.

The "`Manifest-scheme`" property indicates the specification name and version to which manifest files used in the Dflat conform.

The "`Full-scheme`" property indicates the specification name and version of the convention used for fully-instantiated representations of object versions.

The "`Delta-scheme`" property indicates the specification name and version of the convention used for delta-compressed representations of object versions.

The "`Current-scheme`" property indicates the mechanism used to point to the current version directory.

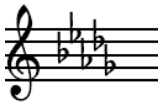
The "`Class-scheme`" property indicates the specification name and version of the convention used for managing the administrative properties of managed objects.

3.5 Lock file (lock.txt)

The OPTIONAL lock file "`lock.txt`" indicates that the Dflat is being processed in a manner that may temporarily 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 Dflat in a manner that affects the state of its home directory



MUST first obtain a write lock on that directory. Any process holding a write lock on a Dflat home directory SHOULD release it.at the earliest safe moment. All processes reading a Dflat 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.

3.6 Log directory (log/)

The OPTIONAL "`log/`" sub-directory holds non-versioned log information about the use of the object.

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

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

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

The sub-directory SHOULD contain a file named "`last-fixity.txt`" containing the date/timestamp in fully-qualified W3C form of the last fixity verification of the object and an identifier of the fixity process, for example:

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

The sub-directory MAY contain a file named "`lock.txt`" that conforms to the syntax, semantics, and normative obligations defined in Section § **3.5**.

3.7 Version directories (v<*nnn*>/)

The REQUIRED sub-directory "`v001/`" holds a representation of the initial version of the object. Additional OPTIONAL sub-directories of the form "v<nnn>/" hold representations of subsequent object versions. The numerical portion of version sub-directory names MUST increment by 1 between subsequent versions. For version numbers 1 through 999, inclusive, the directory name is exactly four characters in length and the numerical value is left-padded with 0's as necessary:

```
v001/  v002/ ... v998/  v999/
```

For version numbers greater than 999 the numerical value MUST NOT be left-padded with 0's and the directory name length will vary as minimally necessary.

```
v1000/  v1001/ ... v9999/ v10000/ ... v99999/ v100000/ ...
```

The home directory file "`current.txt`" SHALL contain the name of the sub-directory holding the current object version, for example:

```
current.txt                          # contains: "v003"
```

```
v001/                                  # version 1
v002/                                  # version 2
v003/                                  # version 3 (current)
```

A version sub-directory MAY contain a file named "`lock.txt`" that indicates a write operation is in-process and that the version is in a temporarily unknown, inconsistent, or incomplete state.

```
v<nnn>/
     [ lock.txt ]
```

The sub-directory MAY contain a file named "`lock.txt`" that conforms to the syntax, semantics, and normative obligations defined in Section § **3.5**.

Each version sub-directory holds a representation of the digital object at a point in time. Version representations MUST be in one of three forms:

- An empty representation;
- A fully-instantiated representation; or
- A delta-compressed representation.

3.7.1 Empty representation (empty.txt)

An empty representation is used in the rare case where no files are associated with the digital object at a point of time.

```
v<nnn>/
       empty.txt
     [ lock.txt ]
```

The file "`empty.txt`", which MUST contain the string "`empty`", serves as an explicit marker of the empty condition.

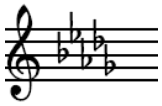
3.7.2 Fully-instantiated representation (full/)

A fully-instantiated representation contains all of the individual files that represent the digital object at a point of time.

```
v<nnn>/
       full/                           # Dnatural directory
     [ lock.txt ]                      # write lock
     [ manifest.txt ]                  # version manifest
```

The individual files that make up the representation are all found in or beneath the "`full/`" sub-directory. The specification to which the "`full/`" directory conforms SHOULD be indicated by the "`Full-scheme`" property of the Dflat properties file "`dflat-info.txt`". The Dnatural specification, described in section § **4** below, MAY be used for this purpose.

The complete set of files and directories that make up the representation SHOULD be listed in the OPTIONAL version manifest file "`manifest.txt`". If present, the specification to which the manifest file conforms SHOULD be indicated by the "`Manifest-scheme`" property of the Dflat



properties file "`dflat-info.txt`". The Checkm [Checkm] specification MAY be used for this purpose.

The Checkm format is:

```
<pathname> <type> <digest> <size> <modtime>
...
```

where "<*pathname*>" is the file or directory pathname relative to the "`full/`" directory; "<*type*>" is the name of a message digest algorithm, or the value "`dir`" in the case of a directory; "<*digest*>" is the message digest value encoded as a hexadecimal string, or "`-`" in the case of the directory; "<*size*>" is the size of the file in octets, or "`0`" in the case of a directory; and "<*modtime*>" is the last modification date/timestamp of the file or directory in fully-qualified W3C form. For example, the file "`full/data/xyz.pdf`" would be referenced in "`manifest.txt`" as:

```
data/xyz.pdf <type> <digest> <size> 2009-07-06T11:41:27+0800
```

The current version of the digital object stored in a Dflat MUST be a fully-instantiated representation.

3.7.3 Delta-compressed representation (delta/)

A delta-compressed representation contains information needed to re-instantiate all of the individual files that represent the digital object at a point of time.

```
v<nnn>/
     [ d-manifest.txt ]            # delta directory manifest
       delta/                      # ReDD directory
     [ lock.txt ]                  # write lock
     [ manifest.txt ]              # version manifest
```

If present, the OPTIONAL delta directory manifest file "`d-manifest.txt`" MUST list all of the files found in the "`delta`/" directory. The specification to which the delta manifest file conforms SHOULD be indicated by the "`Manifest-scheme`" property of the Dflat properties file "`dflat-info.txt`". If the manifest conforms to the Checkm specification, "<*pathname*>" is the file pathname relative to the "`delta/`" directory. For example, the delta file "`delta/add/data/xyz.pdf`" would be referenced in "`d-manifest.txt`" as:

```
add/data/xyz.pdf <type> <digest> <size> 2009-07-06T11:39:02+0800
```

The REQUIRED sub-directory "`delta/`" holds all information necessary to re-instantiate the version. The re-instantiation process MAY assume the prior existence of the immediately previous or subsequent version. The specification to which the "`delta/`" directory conforms SHOULD be indicated by the "`Delta-scheme`" property of the Dflat properties file "`dflat-info.txt`". The ReDD specification [ReDD] , briefly described in section § **5** below, MAY be used for this purpose.

The complete set of version files that would be present after re-instantiation SHOULD be listed in the OPTIONAL version manifest file "`manifest.txt`". If the manifest conforms to the Checkm specification, "<*pathname*>" is the re-instantiated file pathname relative to its imputed parent directory. For example, the re-instantiated file that corresponds to the fully-instantiated file

"full/data/xyz.pdf" would be referenced in "manifest.txt" as:

```
data/xyz.pdf <type> <digest> <size> 2009-07-06T11:41:27+0800
```

All non-current versions of the digital object stored in a Dflat SHOULD be delta-compressed representations.

4 Dnatural

Dnatural is a subsidiary file system convention for representing digital object state in fully instantiated form.

```
<dnatural>/
          0=dnatural_<version>     # Dnatural Namaste signature
        [ admin/ ]                 # administrative declarations
        [ annotation/ ]            # annotation directory
        [ data/ ]                  # data directory
        [ enrichment/ ]            # enrichment directory
        [ log/ ]                   # log directory
        [ metadata/ ]              # metadata directory
```

A Dnatural directory MUST contain a file named "0=dnatural_<*version*>" that is its Namaste signature.

The directory MAY contain a sub-directory named "admin/" that holds versioned administrative declarations about the object stored in the Dflat.

The directory MAY contain a sub-directory named "annotation/" that holds object data or metadata originating from object consumers.

The directory SHOULD contain a sub-directory named "data/" that holds object data originating from the object's producer or curator.

The directory MAY contain a sub-directory named "enrichment/" that holds object data or metadata originating from automated processes of the managerial infrastructure in which the Dflat exists.

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

The directory SHOULD contain a sub-directory named "metadata/" that holds object metadata originating from the object's producer or curator.

5 ReDD

ReDD is a specification for file-level reverse delta compression of versioned sets of files.

```
<redd>/
       0=redd_<version>            # ReDD Namaste signature
     [ add/ ]                      # additions, relative to subsequent
     [ delete.txt ]                # deletions, relative to subsequent
```

The OPTIONAL "`add/`" directory contains the files that need to be added and the OPTIONAL "`delete.txt`" file lists the files that need to be deleted *relative to the subsequent version* in order to fully re-instantiate the target version.

A variant form of the ReDD directory MUST be used if the version entails no changes *relative to the subsequent version*.

```
<redd>/
       0=redd_<version>
       no-change.txt
```

The file "`no-change.txt`", which MUST contain the string "`no-change`", serves as a marker of the no-change condition.

6 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 and arbitrary directory and file names, such as POSIX [POSIX].

To facilitate interoperation it is RECOMMENDED that none of the files found in any of the Dnatural sub-directories have a filename length greater than 255 characters.

7 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>/
           [ 0=dflat_<version> ]
           [ admin/
                [ lock.txt ]
                [ summary-stats.txt ] ]
           [ current.txt ]
           [ dflat-info.txt ]
           [ lock.txt ]
           [ log/
               [ last-access.txt ]
               [ last-fixity.txt ]
               [ lock.txt] ]
             v001/
           [ v002/
             ...  ]
```

The production rule for the Dflat Namaste signature file "`0=dflat_<version>`" is:

```
<dflatsig>    = "0=dflat_" <version> EOL
<version>     = NONNEG 0*("." 1*DIGIT)
NONNEG        = "0" / (1*POSDIG 0*DIGIT)
POSDIG        = "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9"
EOL           = CR / CRLF / LF
```

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
```

More specific rules MAY be defined by each 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 summary statistics file "`summary-stats.txt`" are:

```
<stats>       = 1*<stat>
<stat>        = (<versions> / <files> / <size>) EOL
<versions>    = "Version-count:" 1*WSP NONNEG
<files>       = "File-count:" 1*WSP NONNEG
<size>        = "Total-size:" 1*WSP NONNEG
```

The production rules for the current version file "current.txt" are:

```
<current>     = "v" <small> / <large>
<small>       = 3DIGIT   ; the value "000" is disallowed
<large>       = 4*DIGIT  ; leading zeros are disallowed
```

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

```
<properties>  = 1*<property>
<property>    = (<object> / <manifest> / <full> / <delta> /
                <current> / <class>) EOL
<object>      = "Object-scheme:" 1*WSP <scheme>
<manifest>    = "Manifest-scheme:" 1*WSP <scheme>
<full>        = "Full-scheme:" 1*WSP <scheme>
<delta>       = "Delta-scheme:" 1*WSP <scheme>
<current>     = "Current-scheme:" 1*WSP "file"
<class>       = "Class-scheme:" 1*WSP <scheme>
<scheme>      = <name> "/" <version>
<name>        = 1*VCHAR
```

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

```
<lock>        = "Lock:" 1*WSP <date-time> 1*WSP <process> EOL
<date-time>   = <date> "T" <time>
<date>        =  <year> "-" <month> "-" <day>
```

```
<year>        = 4DIGIT
<month>       = 2DIGIT  ; normal constraints apply, 01 through 12
<day>         = 2DIGIT  ; normal constraints apply, 01 through 31
<time>        = <hour> ":" <minute> ":" <second> "+" <zzzz>
<hour>        = 2DIGIT  ; normal constraints apply, 00 through 23
<minute>      = 2DIGIT  ; normal constraints apply, 00-59
<second>      = 2DIGIT  ; normal constraints apply, 00-59
<zzzz>        = 4DIGIT  ; normal constraints apply, 0000 through 2300
<process>     = 1*VCHAR
```

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

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

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

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

It is RECOMMENDED that only a single white space character (WSP) is used to demarcate the date/timestamp and process identifier.

The structure of a version directory is, if empty:

```
v<nnn>/
       empty.txt
     [ lock.txt ]
```

or, if fully-instantiated:

```
v<nnn>/
       full/
     [ lock.txt ]
     [ manifest.txt ]
```

or, if delta-compressed:

```
v<nnn>/
     [ d-manifest.txt ]
       delta/
     [ lock.txt ]
     [ manifest.txt ]
```

The production rule for the empty version marker file "`empty.txt`" is:

```
<empty>       = "empty" EOL
```

The production rules for a Checkm manifest file "`manifest.txt`" or "`d-manifest.txt`" are:

```
<manifest>    = 1*<record>
<record>      = <pathname> 1*WSP <type> 1*WSP <digest> 1*WSP
                <size> 1*WSP <modtime> EOL
<pathname>    = 1*PCHAR 0*(SEPARATOR 1*PCHAR)
```

```
<type>        = "Adler-32" / "CRC-32"  / "MDs" / "MD5" / "SHA-1" /
                "SHA-256"  / "SHA-384" / "SHA-512" / "dir"
<digest>      = 1*(HEXDIG / "a" / "b" / "c" / "d" / "e" / "f") / "-"
<size>        = NONNEG
<modtime>     = <date> "T" <time>
PCHAR         = <file-system-dependent-path-character>
SEPARATOR     = <file-system-dependent-path-separator>
```

It is RECOMMEDED that only a single white space character is used to demarcate the manifest record fields. It is RECOMMENDED that message digest values use either all upper-case or all lower-case hexadecimal characters.

The structure of a Dnatural directory is:

```
<dnatural>/
           0=dnatural_<version>
         [ admin/ ]
         [ annotation/ ]
         [ data/ ]
         [ enrichment/ ]
         [ log/ ]
         [ metadata/ ]
```

The production rule for the Dnatural Namaste signature file "0=dnatural_<version>" is:

```
<dnaturalsig> = "0=dnatural_" <version> EOL
```

The structure of a ReDD directory is:

```
<redd>/
       0=redd_<version>
     [ add/ ]
     [ delete.txt ]
```

or, if the version entails no change:

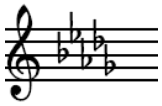
```
<redd>/
       0=redd_<version>
       no-change.txt
```

The production rule for the Dnatural Namaste signature file "0=dnatural_<version>" is:

```
<reddsig>     = "0=redd_" <version> EOL
```

The production rule for the ReDD no-change marker "no-change.txt" is:

```
<no-change>   = "no-change" 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.

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

[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>.

[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>.