DNA Data Storage Sector One

Version 1.0

ABSTRACT: This specification defines the recommended method and embodiment for storing archive metadata within a DNA data storage archive for the purpose of enabling an archive reader to read the archive and then consume the logical structure and its data contents.

This document has been released and approved by SNIA. SNIA believes that the ideas, methodologies, and technologies described in this document accurately represent SNIA goals and are appropriate for widespread distribution. Suggestions for revisions should be directed to https://www.snia.org/feedback/.

SNIA Standard

November 11, 2023

USAGE

DISCLAIMER

The information contained in this publication is subject to change without notice. SNIA makes no warranty of any kind with regard to this specification, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. SNIA shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this specification.

1 Overview

Deoxyribonucleic acid (DNA), when used as a data storage media, lacks key properties found in traditional data storage media (e.g., tape, SSD, HDD). DNA data storage is commonly based on short strings of DNA called oligonucleotides (oligos) that are mixed together without a specific physical ordering scheme, whereas traditional data storage media types are based on magnetic media (e.g., hard disk drives and tape) or silicon microchips (e.g., NAND). DNA storage media lacks a dedicated and integrated controller serving as a natural point of interconnect and interface between the consuming system and the media. Further, DNA storage media lacks an organizational means by which proximity of one media subcomponent can be understood in relation to another media subcomponent.

The specification for DNA data storage sector zero should be considered a prerequisite to understanding and consuming this specification. Sector zero enables an archive reader to understand two key attributes: who wrote the archive, and what CODEC was used to write the sector one metadata.

With an understanding of sector zero, an archive reader has an understanding of who wrote the archive and what CODEC to use to read sector one. Using this CODEC and following this specification will enable the archive reader to understand the sector one contents, that is, the logical layout of the archive and how to access the data contents therein.

2 Sector Zero vs Sector One

To provide clarity and context, sector zero is intended to provide readers with the ability to access sector one. Sector zero indicates who wrote the archive, and with which CODEC, and sector one provides details about the logical structure of the archive and the CODEC used to read the archive contents.

The idea behind this division comes from the need to represent some starter data without the need of a CODEC. The amount of data represented in sector zero is small and fits into a single oligonucleotide. Sector one, on the other hand, contains a much richer set of data, extending beyond the capacity of a single oligo. Given that no facilities exist to specify an oligo order to accurately reassemble the payload from each oligo's contents, the CODEC must assemble the oligos correctly and decode them into readable data.

3 Assumptions

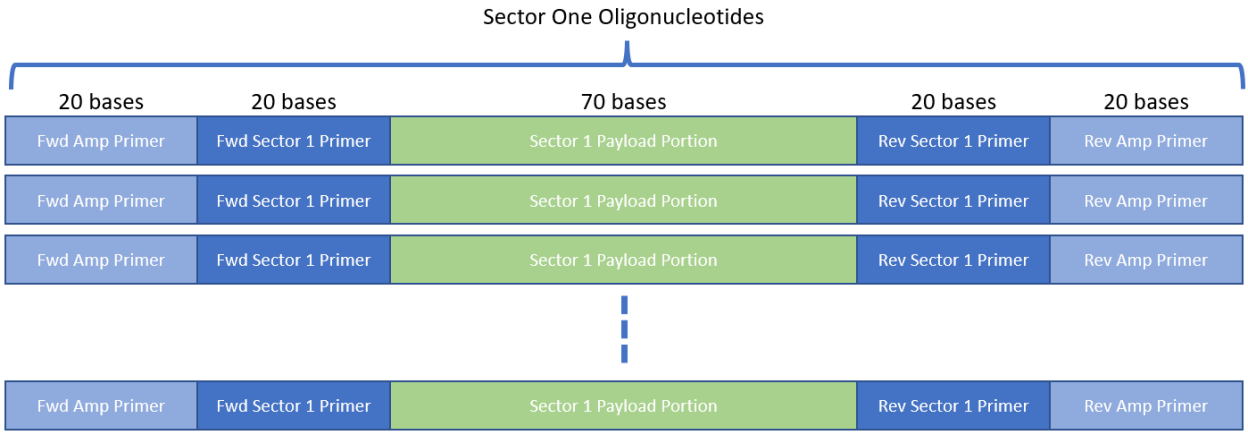
The intent of sector one is to supply the archive reader with an understanding of how to access the logical layout of the archive's data contents and details necessary to enable consumption of those contents. The following are the assumptions an archive reader and writer should make in following this process.

- It is assumed that each oligonucleotide contains exactly 150 bases.

- It is assumed that sector one contents will span multiple oligonucleotides and will require use of a CODEC to decode the payload contained therein.

- It is assumed that the sector one oligonucleotides will be comprised only of bases found in natural DNA, that is adenine (A), cytosine (C), guanine (G), and thymine (T).

- It is assumed that the sector one payload will follow the schema and structure defined later in this specification.



4 Sector One Schema

The contents of sector one shall be stored in a JavaScript Object Notation (JSON) object with the following constraints:

- When serializing the object, null values shall be excluded from serialization.

- Non-pretty serialization should be used to minimize space consumption.

The sector one object is hierarchical in its structure, that is, it is an object at the top level, containing key-value pairs where the value may be discrete (e.g., a string, Boolean, etc.) or another object.

Properties within the sector one object are annotated as (R)equired, (O)ptional, or (W)riter-managed:

- Required properties must be present both within the JSON object persisted in the archive as well as any external representation of the sector one data; additionally, a non-null and non-default value must be specified.

- Optional properties are recommended but not required.

- Writer-managed properties can be populated at the discretion of the archive writer with the understanding that key names will not conflict with key names assigned by the specification.

Values must conform to one of the following types:

- Object (o), i.e., a dictionary of key-value pairs where the value is discrete, an array, or another type
- String (s), represented in UTF-8
- Number (n)
- Float (f)
- Boolean (b), with value in lowercase and not encapsulated in quotation marks
- Identifier (i), i.e. a UID, UUID, or GUID, represented as a string
- Timestamp (t) in ISO-8601 form, including the date, represented as a string.

The tables that follow outline the fields, key names, disposition (required, optional, writer-managed), description, data type, and maximum recommended lengths.

4.1 Top-level

The top-level object shall have the following schema, semantics, and properties.

Name	Key	Disposition	Description	Type	Max Len
Top-level object					
ID	id	R	Universal identifier for the archive	i	64
Description	desc	O	Description of the archive contents	s	64
Hash Details	hash	O	Hash details	o	
Sequencing Details	seq	R	Sequencing details	o	
CODEC Details	codec	R	CODEC details	o	
Timestamp	ts	O	Timestamp from archive creation	t	32
Writer Details	vendor	O	Archive writer details	o	
Writer ID	vendorid	R	Archive writer vendor ID, from sector zero table	s	64
Optional Fields	opt	O	Optional fields	o	
Additional Parameters	addl	W	Additional parameters	o	

If the archive writer has opted to not register their vendor information to receive a vendor ID, the archive writer should use the following value:

- `ACGACACAGTGATCATGCAGTCTCTATAGAGATCT`

4.2 *Hash Object*

The hash object, found within the `hash` key in the top-level object, shall have the following schema, semantics, and properties.

Name	Key	Disposition	Description	Type	Max Len
Hash details object					
Algorithm	alg	O	Algorithm (e.g. SHA 256, SHA 512)	s	16
Hash Value	val	O	Hash value (note: leaving room for quantum-safety)	s	256
Additional Parameters	addl	W	Additional parameters	o	

The hash object is optional; it is up to the archive writer to determine its scope and usage. The use of a CODEC for writing data into the archive, including both sector one metadata and archive data contents, implies that a meaningful and effective level of error correction, error detection, and error recovery are present.

Although it is an optional object, the governing body recommends that all archive writers follow two primary approaches for providing and validating integrity.

- Use the top-level `hash` object in the following way:
    - JSON serialize and minify the entirety of the sector one object **without the presence of the hash object or its key**
    - Calculate the hash against the minified JSON string
    - Populate the hash object accordingly
- Ensure hashing capabilities are integrated within the logical structure of the archive contents, e.g. within the filesystem (or other representation) contained within the archive

Further, when validating hashes, it is recommended that the reader perform the following steps:

- Ensure the JSON object is minified
- Take note of the hash object value, and then remove the hash object and its key from the JSON object
- Calculate the hash against the minified JSON string

- Ensure that the calculated hash value matches the original value that was removed

4.3 Sequencing Details Object

The sequencing details object, found within the `seq` key in the top-level object, shall have the following schema, semantics, and properties. The goal of the sequencing details object is to provide relevant information to the sequencer for more efficient sequencing.

Name	Key	Disposition	Description	Type	Max Len
Sequencing details object					
Oligo Size Minimum	osizemin	R	Minimum size of oligonucleotides (int32)	i	4
Oligo Size Median	osizemedian	R	Median size of oligonucleotides (int32)	i	4
Oligo Size Maximum	osizemax	R	Maximum size of oligonucleotides (int32)	i	4
Oligo Replication Count Minimum	orepmin	R	Minimum number of replicated oligos	i	4
Oligo Replication Count Maximum	orepmax	R	Maximum number of replicated oligos	i	4
Oligo Count	ocount	R	Number of oligonucleotides (int64)	i	8
Universal Leading Adapter	ulead	R*	Universal leading adapter	s	32
Universal Tailing Adapter	utail	R*	Universal tailing adapter	s	32
Adapter Presence	adapter	R*	Flag indicating if adapters are present	b	
Natural Bases	natural	R	Flag indicating if natural DNA bases are used	b	
Coverage Information	coverageinfo	R	Recommended coverage information	s	
Additional Parameters	addl	W	Additional parameters	o	

* Leading `ulead` and tailing `utail` adapters must be specified unless `adapter` is set to `false`. In cases where `adapter` is `false`, library preparation will be required. In cases where adapter is `true`, `ulead` and `utail` must be included.

4.4 CODEC Details Object

The CODEC details object, found within the `codec` key in the top-level object, shall have the following schema, semantics, and properties.

Name	Key	Disposition	Description	Type	Max Len
CODEC details object					
CODEC ID	id	R	Alliance-assigned identifier for the CODEC	s	64
CODEC Vendor	vendor	O	Vendor authoring the CODEC	s	32
CODEC Name	name	O	Name of the CODEC	s	32
CODEC Version	ver	O	Version of the CODEC	s	16
CODEC URI	uri	R	Uniform resource identifier (URI), e.g. location	s	128
CODEC Parameters	params	R, W	Parameters for the CODEC	o	
Additional Parameters	addl	W	Additional parameters	o	

If the archive writer has opted to not register their CODEC information to receive a CODEC ID, the archive writer should use the following value:

- `ATAGTCTCTGATCACTCACGTATGTGCGTGAGCTA`

4.5 *CODEC Parameters Object*

The CODEC parameters object, found within the `params` key in the `codec` object, enables the transfer of CODEC specific information to the decoder/sequencer. This will be vendor specific based on the CODEC being used and thus its fields are not defined and will be decided by the synthesis/CODEC vendor. The `params` property is required, but its contents will be decided by the vendor.

4.6 *Writer Vendor Object*

The writer vendor object, found within the `vendor` key in the top-level object, shall have the following schema, semantics, and properties.

Name	Key	Disposition	Description	Type	Max Len
Vendor details object					
Vendor Name	name	O	Name of the vendor providing the archive	s	32
Vendor URI	uri	O	URI for the vendor	s	128
Vendor Contact Information	contact	O	Contact information for the vendor	s	128

4.7 *Optional Parameters Object*

The optional parameters object, found within the `opt` key in the top-level object, shall have the following schema, semantics, and properties.

Name	Key	Disposition	Description	Type	Max Len
Optional fields object					
MIME Type	mime	O	MIME type of the archive contents	s	64

4.8 *Additional Parameters Object*

The additional parameters object, found within the `addl` key in the top-level object and other objects, shall have a schema, semantics, and properties as defined by and at the discretion of the archive writer.

5 Example Sector One JSON

Following the requirements described in the previous section, an example sector one JSON object is presented both in its non-minified form (only useful for readability) and minified form (how it would be represented in the archive).

5.1 *Non-Minified Sector One JSON*

The following is an example non-minified JSON of sector one.

```
{
  "id": "b055d80b-ab83-4e10-b287-990864ece972",
  "desc": "A giant zip file of all your stuff"
  "hash": {
    "alg": "SHA256",
    "val": "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad",
    "addl": {
      "key": "value"
    }
  },
  "seq": {
    "osizemin": 150,
    "osizemedian": 150,
    "osizemax": 150,
    "orepmin": 10,
    "orepmax": 1000,
    "ocount": 9173260918,
    "ulead": "ACTG",
    "utail": "GTCA",
    "natural": true,
    "coverageinfo": "...",
    "addl": {
      "key": "value"
    }
  },
  "codec": {
    "id": 1234,
    "vendor": "Vendor Name",
    "name": "Default CODEC",
    "ver": "1.6.12",
    "uri": "http://foo.bar/baz",
    "params": {
      "key": "value"
    },
    "addl": {
      "key": "value"
    }
  },
  "ts": "2023-03-14T16:57:32+00:00",
  "vendor": {
    "name": "Dell Technologies",
    "uri": "https://dell.com",
    "contact": "dnaarchives@dell.com, +1-512-555-1212"
  },
  "vendorid": "ACTGACTG…",
  "opt": {
    "mime": "application/zip"
  },
  "addl": {
    "key": "value"
  }
}
```


5.2 Minified Sector One JSON

The following is an example minified JSON of sector one.


```
{"id":"b055d80b-ab83-4e10-b287-990864ece972","desc":"A giant zip file of all your
stuff","hash":{"alg":"SHA256","val":"ba7816bf8f01cfea414140de5dae2223b00361a396177a9
cb410ff61f20015ad","addl":{"key":"value"}},"seq":{"osizemin":150,"osizemedian":150,"
osizemax":150,"orepmin":10,"orepmax":1000,"ocount":9173260918,"ulead":"ACTG","utail"
```

```
:"GTCA","natural":true,"coverageinfo":"...","addl":{"key":"value"}},"codec":{"id":12
34,"vendor":"Vendor Name","name":"Default
CODEC","ver":"1.6.12","uri":"http://foo.bar/baz","params":{"key":"value"},"addl":{"k
ey":"value"}},"ts":"2023-03-14T16:57:32+00:00","vendor":{"name":"Dell
Technologies","uri":"https://dell.com","contact":"dnaarchives@dell.com, +1-512-555-
1212"},"vendorid":"ACTGACTG…","opt":{"mime":"application/zip"},"addl":{"key":"value"
}}
```


6 Reading Sector One

Generally, sector zero and sector one should only need to be read in cases where the external representation (see section 1.7) of these properties is not available. In many cases, archive owners will understand a priori the contents, CODEC used, and other metadata, as this information is likely stored in an external archival management system.

In cases where sector one needs to be read from DNA, use the following process:

1. Amplify sector zero using sector zero dedicated primers (refer to the sector zero specification)
2. Amplify sector one using sector one dedicated primers.
3. Obtain the CODEC as described in the sector zero payload.
4. Decode sector one using the CODEC described in the sector zero payload.

Note that the diagrams above indicate the presence of a forward and reverse amplifying primer. These should only be used in cases where you need to amplify not only the sector one contents, but also the primers that encapsulate those contents. The amplifying primer for sector one may not be unique to sector one.

For sector one, the following primers should be used. These primers have been selected to mitigate common issues related to GC content, homopolymers, self-dimers, and homo-dimers. These primers should not be used anywhere else in the archive; they should be universally unique.

- Forward primer: ACAACGCTCAAATCAGGG
- Reverse primer: GTGACGGTAGGTGAAATC

7 External Representation of Sector One

The contents of sector 1 should be readable from outside of the archive and must be readable from inside the archive. Given that the physical constraints of containers holding the media are not yet well-defined, the choice of embodiment to make sector 1 readable outside of the archive is at the discretion of the archive writer. Standard mechanisms such as QR codes and data matrices may be appropriate in environments that have ample real estate given the size constraints of these mechanisms. In other cases, technologies such as near-field communication may be more appropriate. Thus, the specification does not place a burden on those designing DNA data storage systems; rather this specification allows them to choose the embodiment that is

appropriate for their systems based on size constraints and other parameters they deem important.

8 Summary

The embodiment defined in this specification enables archive writers to store metadata within the archive to service a future archive reader, specifically providing them with the parameters necessary to consume the archive related to the CODEC, sequencer, logical layout, and others. This is referred to as sector one.

Sector one serves as a starting point for archive readers to understand the logical layout of the archive and how to read its payload. Sector one is represented as a minified JSON object, encoded using a CODEC, and written to the archive using multiple oligonucleotides that contain only natural DNA bases.

Having the contents of sector one affords the archive reader with an understanding of the logical layout of the archive's contents (e.g. filesystem, object, or other) and the parameters needed for the sequencer and CODEC to then be able to consume the contents therein.