Overview of SnapGene File Format
May 9, 2013
SnapGene Version 1.5

Reading SnapGene Files
SnapGene uses a "segmented" binary file format, which means that a .dna file is a series of file segments. Each file segment consists of:

- a single byte that encodes the segment type
- 4 bytes that encode N = the length in bytes of the subsequent data segment
- N bytes of segment data

Thus, if you want to read the DNA sequence, walk through the file looking for the DNA segment. Skip past each unwanted segment by reading the 5-byte header and then jumping ahead by the segment data length. As this example illustrates, it is not necessary to understand all parts of a SnapGene file in order to read a subset of the information.

SnapGene .dna files store information about the DNA, features, primers, notes, history, display options, and more. This document will not cover the file segments that are typically irrelevant for applications other than SnapGene. For example, rapid searching of a .dna file is facilitated by indexes, including a MICA index (MICA: desktop software for comprehensive searching of DNA databases. Stokes WA, Glick BS. BMC Bioinformatics. 2006 Oct 3;7:427). Here, we will describe the following file segments:

File Description
DNA
Additional Sequence Properties
Features
Primers
Notes

File segments can be encountered in any order, except that the File Description segment will always be listed first. In general, a file segment should exist only once. The exception is History Nodes (ancestral sequences), which are not described here.

File Description Segment
The file description segment serves mainly to provide a “magic cookie” at the top of a file. A typical file description might look like this:

```
09              // type 9 = File Description segment
00 00 00 0E     // data length = 14 bytes
"SnapGene"      // The segment type + the length + "SnapGene" encoded in ASCII make
                // up the magic cookie that can be used to identify a SnapGene file.
00 01           // file type: 0 = unknown, 1 = DNA
00 0A           // export version #: This value is used only by SnapGene.
00 04           // import version #: This value is used to determine if the file was created with a newer version
                // of SnapGene and thus may be difficult for the current application to read.
```

DNA Segment
A DNA segment encodes the full top strand sequence, including any “ghost” bases that represent the complement of bottom strand overhangs. The DNA can include both upper- and lowercase characters. A typical DNA segment might look like this:

```
00              // type 0 = DNA
00 00 15 4C     // 5452 bytes = sequence length (5451 bp) + 1 byte for sequence properties
1F              // sequence properties byte, see below
```

"TGCGGT…" // the ASCII DNA sequence
The sequence properties byte is used to encode the following sequence attributes:

Bit 1: topology	1 = circular, 0 = linear
Bit 2: strandedness	1 = double-stranded, 0 = single-stranded
Bit 3: Dam methylated	1 = true, 0 = false
Bit 4: Dcm methylated	1 = true, 0 = false
Bit 5: EcoKI methylated	1 = true, 0 = false

Bits 6-8 are not currently used.

The remaining file segments described below all have their data sections encoded in XML.

Additional Sequence Properties

This segment is used to encode additional properties, primarily information about linear sequence endpoints. A typical segment might look like this:

```
08              // type 8 = Additional Sequence Properties
XX XX XX XX     // segment data length
"<AdditionalSequenceProperties>
<OriginalFirstBase>514</OriginalFirstBase>
<UpstreamStickiness>-1</UpstreamStickiness>
<DownstreamStickiness>-1</DownstreamStickiness>
<UpstreamModification>Unmodified</UpstreamModification>
<DownstreamModification>Unmodified</DownstreamModification>
</AdditionalSequenceProperties>"
```

The XML format consists of a AdditionalSequenceProperties tag wrapping one or more other tags such as:

UpstreamEnzymeName: If a linear sequence was excised from a parental sequence using restriction enzymes, this name encodes the enzyme that generated the upstream end of the current sequence.

UpstreamStickiness: Is the upstream end sticky? 0 = blunt, +2 = 2-base 5' overhang, -3 = 3-base 3' underhang

UpstreamModification: Is the upstream end unmodified ("Unmodified"), covalently closed ("CovalentlyClosed"), or 5' phosphorylated ("FivePrimePhosphorylated")? If this tag is absent, the default is “Unmodified”.

Similar attributes for the downstream end of the DNA sequence are:

DownstreamEnzymeName
DownstreamStickiness
DownstreamModification

Other tags that you may encounter include:

CountBasesFrom: What is the number of the first base in the sequence? If this tag is absent, the default is +1.

OriginalFirstBase: Which base was originally the numerical origin? This value is used to restore the numerical origin when a sequence is linearized and then recircularized, e.g., during TA cloning.

Features Segment

This segment is used to encode all feature data. A typical segment might look like this:

```
0A                  // type 10 = features
XX XX XX XX   // segment data length
"<Features …> …  </Features>"
```

The features section consists of a pair of "<Features>" tags wrapping one or more "<Feature>" tags, one for each feature in the sequence. If the sequence has no features, this file segment is absent. A typical feature might be encoded like this:

```
<Feature name="ScARG4" directionality="1" translationMW="52004.39" type="CDS"
consecutiveTranslationNumbering="1">
  <Segment range="2702-4093" color="#ff8000" type="standard" translated="1"/>
  <Q name="codon_start"><V int="1"/></Q>
  <Q name="transl_table"><V int="1"/></Q>
</Feature>
```

Here, the “Q” tag represents a qualifier and the “V” tag represents a qualifier value.

A Feature tag can contain one or more attributes for feature properties such as:

name: the feature name in UTF8

directionality: 0 = nondirectional, 1 = forward directional, 2 = reverse directional, 3 = bidirectional

geneticCode: GenBank genetic code ID

translateFirstCodonAsMet: Should first amino acid be translated as Methionine? 1 = true, 0 = false

translationMW: MW in daltons of the polypeptide encoded by a translated feature

type: GenBank feature type, e.g., "CDS"

cleavageArrows: string encoding location of cleavage arrows within the feature as a list of numbers joined by commas, where a cleavage arrow before the first base in the DNA sequence is numbered 0 even if CountBasesFrom (see above) has a different value

readingFrame: the reading frame that should be used when translating the feature, where possible values include [1,2,3,-1,-2,-3]

visible: Is the feature visible in views outside of Features view? 1 = visible, 0 = hidden

Each feature is represented as consisting of one or more segments, and each segment is encoded using a "Segment" tag. Feature segments encode feature locations, where the first base in the DNA sequence is numbered +1 even if CountBasesFrom (see above) has a different value. For example:

```
<Segment range="2702-4093" color="#ff8000" type="standard" translated="1"/>
```

specifies a segment from bases 2702 to 4039, with a hex color code of #ff8000, and with the segment included in the feature translation.

A segment of type “standard” is displayed as part of the feature, and a segment of type “gap” is displayed as a dashed line. Introns are represented using the “gap” segment type.

Each GenBank qualifier is encoded using a "Q" tag wrapped around one or more "V" tags. Qualifiers have GenBank-style names (e.g. "codon_start"), and qualifier values can encode data in a variety of formats depending on the particular qualifier.

Primers Segment
This segment is used to encode all primer data. A typical segment might look like this:

```
0A                  // type 5 = primers
XX XX XX XX   // segment data length
"<Primers …> …  </Primers>"
```

The "Primers" tag itself does not encode much useful information. All primer data are stored within individual "Primer" tags that might look like this:

```
<Primers>
  <HybridizationParams minContinuousMatchLen="10" allowMismatch="1" minMeltingTemperature="40"/>
  <Primer name="Cse4 CT For" sequence="ctgatcGAATTCCAGATACAAACCTGGAACG"
description="&lt;html&gt;&lt;body&gt;&lt;/body&gt;&lt;/html&gt;">
    <BindingSite location="4431-4455" boundStrand="0"
annealedBases="GAATTCCAGATACAAACCTGGAACG" meltingTemperature="58">
      <Component bases="ctgatc"/>
      <Component hybridizedRange="4431-4455" bases="GAATTCCAGATACAAACCTGGAACG"/>
    </BindingSite>
  </Primer>
  …
</Primers>
```

The "HybridizationParams" tag is used to encode the hybridization options for the given .dna file, and should be unnecessary for conversion to other file formats.

A given "Primer" encodes the primer name (as a UTF8 string), the primer sequence (as an ASCII string), and a description of the primer (as a UTF8-encoded HTML string).

A BindingSite tag represents the location where a primer hybridizes, where the first base in the DNA sequence is numbered 0 even if CountBasesFrom (see above) has a different value. (Note that for primers but not features, the locations stored in the file are offset by 1 bp from the numbers displayed by SnapGene.) For a forward primer boundStrand="0", and for a reverse primer boundStrand="1". The meltingTemperature is indicated in °C. Various annealed or nonannealed portions of the primer are represented using Component tags.

Notes Segment
This segment encodes all information specified using the Description Panel. A typical segment might look like this:

```
06                  // type 6 = notes
XX XX XX XX   // segment data length
"<Notes>
<Type>Synthetic</Type>
<ConfirmedExperimentally>0</ConfirmedExperimentally>
<Created>2012.3.26</Created>
<LastModified>2012.3.26</LastModified>
<CreatedBy>Glick Lab</CreatedBy>
<TransformedInto>DH10B</TransformedInto>
</Notes>"
```

In other words, a "Notes" tag wraps one or more other tags. Note that all strings are encoded as UTF8.

Type: "Natural" or "Synthetic"
ConfirmedExperimentally: 1 = true, 0 = false
Description: HTML-encoded description
Created: sequence creation date formated as YYYY.MM.DD
LastModified: sequence modification date formatted as YYYY.MM.DD

AccessionNumber: ASCII string
CodeNumber: ASCII string
CreatedBy: string (corresponds to Sequence Author)
Organism: string
SequenceClass: string
TransformedInto: string
Comments: HTML-encoded description

Finally, if one or more references are present, they are encoded within a pair of "References" tags. For example:

<References>
<Reference title="<html><body>MICA: desktop software for comprehensive searching of DNA databases.</body></html>" pubMedID="17018144" journal="<html><body>BMC Bioinformatics 2006;7:427.</body></html>" authors="<html><body>Stokes WA, Glick BS.</body></html>"/>
</References>

A given Reference encodes the various reference attributes using “title”, “pubMedID”, “journal”, and “authors” tags. All of these tags are HTML-encoded except “pubMedID”, which is standard UTF8.

Writing Files that SnapGene Can Read

As an alternative to creating .dna files, it is much more convenient to create GenBank (.gbk) files with formatting information that can be decoded by SnapGene. Examples of such files can be generated by using the “Export Sequence…” command from the File menu of SnapGene.

Information at the top of the GenBank file is imported into the Description Panel in an obvious way, except that the DEFINITION field is imported into the “Description” box. In addition, SnapGene extends the GenBank format to encode data about feature names, feature directionality, feature segment ranges (for multi-segment features), and feature or segment colors. This information is recorded using extra "/note" qualifiers. To decode the extra “/note” qualifier data correctly, SnapGene must find the following marks in the GenBank file:

a) The locus name must contain "Exported File", e.g.:

```
LOCUS       Exported File           4373 bp ds-DNA    circular SYN 07-MAR-2012
```

b) The last REFERENCE must have the TITLE set to "Direct Submission", and the JOURNAL must begin with "Exported [DATE] from SnapGene", e.g.

```
REFERENCE   2  (bases 1 to 4373)

  AUTHORS   [the Sequence Author is listed here]

  TITLE     Direct Submission

  JOURNAL   Exported Apr 21, 2012 from SnapGene 1.0.0
            http://www.snapgene.com
```

If a file passes both of these tests, SnapGene will attempt to make use of feature formatting information present in the extra “/note” qualifiers. The rules for encoding this information are given below.

1. The name of a feature is encoded in the first “/note” qualifier. For example:

```
/note=”AmpR”
```

2. The feature color, segmentation, and cleavage arrows are indicated in the last “/note” qualifier, as follows.

a) A feature with a single segment is indicated as follows:

```
/note=”color: [FEATURE SEGMENT COLOR]; direction: [FEATURE DIRECTIONALITY]”
```

where [FEATURE SEGMENT COLOR] is in hexadecimal format. If the feature has a line appearance, the feature color is encoded as:

```
#------
```

[FEATURE DIRECTIONALITY] can be one of the following:

```
RIGHT
LEFT
BOTH
```

This direction information and the preceding semicolon are omitted if the feature is nondirectional, or if the directionality is implicit because the feature is translated or has a "/direction" qualifier.

For example, a nondirectional feature with line appearance would be encoded as:

```
/note="color: #------"
```

A bidirectional green feature would be encoded as:

```
/note="color: #00ff00; direction: BOTH"
```

b) A feature with multiple segments is indicated with multiple lines, which collectively are enclosed within a pair of quotes. The first line is one of the following, where [NUMBER] is an integer:

```
This feature has [NUMBER] segments:
This forward directional feature has [NUMBER] segments:
This reverse directional feature has [NUMBER] segments:
This bidirectional feature has [NUMBER] segments:
```

The first variant is used for a nondirectional feature, or for a feature in which the directionality is implicit because the feature is translated or has a "/direction" qualifier.

Subsequent lines are used to encode information about individual non-gap segments. Each segment is encoded using the following format:

```
[SEGMENT NUMBER]: [FIRST BASE] .. [LAST BASE] / [SEGMENT COLOR] / [SEGMENT NAME]
```

where the segment name and preceding backslash are optional. For example:

```
1: 1000 .. 2000 / #ff0000 / Red Segment
```

c) If a cleavage arrow is present between two segments or at an end of the feature, this information is encoded at the end of the last “/note” qualifier, using the following format:

```
Cleavage site after base [BASE NUMBER]
```

or

```
Cleavage sites after bases [BASE NUMBER, BASE NUMBER]
```

For example, the last line of the last “/note” qualifier might read

```
Cleavage sites after bases 5, 16, 200
```

3. Any intervening “/note” qualifiers hold actual notes about the feature.

The next page shows several complete examples of information encoded in “/note” qualifiers.

```
promoter        complement(2200..2230)
                /note="lac promoter"
                /note="promoter for the E. coli lac operon"
                /note="This reverse directional feature has 3 segments:
                   1: 2200 .. 2206 / #ffffff / -10
                   2: 2207 .. 2224 / #ffffff
                   3: 2225 .. 2230 / #ffffff / -35"

primer_bind     complement(2152..2168)
                /note="M13 rev"
                /note="common sequencing primer, one of multiple similar
                variants"
                /note="color: #a020f0; direction: LEFT"

CDS             459..1319
                /codon_start=1
                /gene="bla"
                /product="beta-lactamase"
                /note="AmpR"
                /note="confers resistance to ampicillin, carbenicillin, and
                related antibiotics"
                /note="This feature has 2 segments:
                   1:  459 ..  527 / #ccffcc / signal sequence
                   2:  528 .. 1319 / #ccffcc
                Cleavage site after base 527"
```

History of Format Changes

Version 1.1
Corrected “consequtiveTranslationNumbering” to “consecutiveTranslationNumbering” in the Features segment. Both spellings are now recognized by SnapGene.

Version 1.2
In the Features segment, XML tags and attributes were renamed as follows to save space:

<Qualifier …>	➞	<Q …>
<QualifierValue …>	➞	<V …>

We also simplified some of the QualifierValue (now V) attributes:

booleanVal	➞	bool
yearVal	➞	year
month	➞	month
day	➞	day
rangeVal	➞	range
textVal	➞	text
predefinedVal	➞	predef
intVal	➞	int

The different variants of these names are now recognized by SnapGene.