Proto-IO Version 2

Brian Walenz (bwalenz@jcvi.org)

May 1, 2007

[section about protoIO, describe the basic format, the data types, etc]

Input

The input to the Celera Assembler consists of sequencer reads and relationships between the reads. Two types of relationships are defined: libraries and mates. A library indicates that all reads in this collection come from the same insert library and thus share numerous properties: end orientation, clone size, randomness, approximate read size, etc. A mate indicates that exactly two reads are from opposite ends of a single clone in a library.

Common record fields

Several records use the same record fields with the same meaning. These are described here, instead of in each record.

- The action (act) is one of add, modify, ignore or delete (A, M, I, D, respectively). The action tells the gatekeeper module what to do with the message.
- The comment (com) and source (src) fields store a free-format comment string.
- Lines beginning with the comment character '#' are ignored. Lines inside multi-line record fields (e.g., com:, src:, seq:, qlt:) are never comment lines. An example batch record:

```
{BAT
bna:ExampleBatch
acc:1003949234243
#  This line is a comment and is ignored.
com:
#  This line is stored in the batch record.
.
}
```

Version record (VER)

The version record tells the assembler what version of the input specification should be used to read messages from this point forward. Any number of VER records may be present in a single file; the last version record encountered is in effect. Files without a VER record are assumed to comply with version 1 of the specification.

The writer will always write to the latest specification version.

```
{VER
ver:<version-number>
}
```

Batch record (BAT)

The optional batch record can contain comments on the source of the file.

```
{BAT
bna:<name>
acc:<batchUID>
com:
<comment>
.
}
```

Library record (LIB)

The library record describes properties about an insert library. Each library has an insert size, specified as a mean (mea) and standard deviation (std). The orientation (ori) describes the orientation implied when two fragments are linked; one of innie, outtie, normal or unoriented. A library is:

I	Innie if the 3' ends of mated reads are closest
O	Outtie if the 5' ends of mated reads are closest
N	Normal if that the 3' end of one read is closest to the 5' end of another read. Note that it is not possible to specify that the 3' end of a particular read is closest to the 5' end of another particular read.
U	Unoriented if the library does not support mated reads. Link messages supplied for reads in this library will generate an error.

```
{LIB
act:<action>
acc:<UID of library>
ori:<link orientation>
mea:<mean>
std:<standard deviation>
src:
<free format text, description of the source of this data>
.
nft:<number of optional features>
fea:
<list of optional features>
.
}
```

Any number of optional features may be supplied to a library. These features are of a more experimental nature, are subject to change, and not formalized in the library specification. Additional features supported by a specific version of the CA should be listed in the release notes. For example:

```
nft:3
fea:
#  All reads in this library are not randomly sequenced
isRandom=0
#  All reads in this library are from a 454 instrument
Is454=1
#  Do not apply Overlap Based Trimming to this library
OBT=0
.
```

Features supported by all format 2 versions are:

- doNotTrustHomopolymerRuns
- hpsIsFlowGram
- hpsIsPeakSpacing
- doNotOverlapTrim
- isNotRandom

Fragment record (FRG)

The fragment record is the primary input record type. It contains the sequence, quality values and ancillary data for each fragment to be used in the assembly.

The random fragment flag (rnd) indicates if this read is truly a random read. Its' value is 0 if the read is not random, and 1 otherwise. All libraries are assumed to generate random fragments (unless a library feature tells otherwise), this flag allows specific reads to be isolated as non-random.

The status code (sta) describes the overall health of the read, e.g., good, E. coli contamination, BAC vector, etc.:

G	good read
B	short clear range, some evidence of high signal
U	3730xl sentinel base call of NNNNN, basically nothing there
W	low signal, short clear range, and/or trace tuner failure
X	some signals high, others low, and short clear range
V	BAC vector screening trash
E	*E. coli* contamination screening trash
I	Insert is only poly A/T, or insert is fewer base pairs than the project threshold (usually 100 base pairs)
R	Rearrangement of vector. The clear range of the read following vector trimming includes evidence of vector sequence

The library (lib) is the UID of the insert library this read comes from. The plate (pla) and plate location (loc) provide some level of tracking where this read was sequenced.

The sequence, quality values and homo-polymer run/peak spacing are supplied as multi-line ….. Quality values are encoded in ASCII by adding 48 (ASCII zero) to the quality value. Homo-polymer run/peak spacing data is optional.

There are three clear ranges associated with each fragment, the vector, quality and final clear range. The vector clear range (clv) indicates the portion of the read which is free from known sequencing vector. The quality clear range (clq) indicates the portion of the read which is of high sequencing quality. The final clear range (clr) is a combination of the vector and quality ranges, usually the intersection. Several special cases exist for clear ranges:

1. To indicate that either the vector or quality clear range is not known, omit the clv: or clq: entry.
2. To indicate no sequence is in a clear range, any begin >= end may be used, with 0,0 being the standard. For example, "clr:0,0".

```
{FRG
act:A
acc:<read UID>
rnd:<random read flag>
sta:<status code>
lib:<library UID>
pla:<plate UID>
loc:<plate location code>
src:
<free format text, description of the source of this data>
.
seq:
<sequence>
.
qlt:
<quality values>
.
hps:
<homo-polymer run or peak spacing or etc info>
.
clv:<beg,end>
clq:<beg,end>
clr:<beg,end>
}
```

Linkage records (LKG)

The linkage record contains a pair of fragments that are linked. Each fragment will indicate which library it is from, and the library will describe the links (orientation, type) themselves. Fragments must be from the same library to be linked.

```
{LKG
act:<action>
frg:<fragment-uid>
frg:<fragment-uid>
}
```