Using the SRA Data Block Descriptor

Draft LK ~~18 Nov~~04 Dec 2009

National Center for Biotechnology Information – National Library of Medicine

1 Overview

The SRA schema version 1.1 supports the constructs to help describe the assignment of run file objects to SRA run data blocks. Run DATA_BLOCKs are specifications for the archive loader. Once loaded into the archive, the parameters are no longer needed in order to interpret the data that was archived.

This content will eventually join the SRA XML Writer's Guide.

1.1 Overview of Data Block Descriptor Usage

The SRA Run DATA_BLOCK is intended for use to convey information to archive loaders. Once the data have been loaded into the Archive and converted into an SRA native object, the information in the DATA_BLOCK descriptor is no longer relevant to users of the data.

The DATA_BLOCK is optional in the schema, but is required for all RUN XML documents used for submission. This is so that when RUN XML documents are returned to users of the Archive, or mirrored between Archives, the DATA_BLOCK section can be redacted.

There are two classes of parameters: DATA_BLOCK descriptor attributes and FILE attributes.

1.2 Related Documents

- SRA File Formats Guide (under development)
- SRA XML Specification Release SRA_1-1 Change Notice

2 DATA_BLOCK Descriptor Attributes

2.1 Multiple Data Blocks

The XML schema allows you to specify multiple data blocks in sequence order, but you are not guaranteed to emit the blocks in any order. Use the new **DATA_BLOCK.serial** attribute to impose a total ordering on the data blocks so that they will get loaded in the order specified.

Example: One SOLiD run broken into 95 pieces for ease of transmission:

```
<DATA_BLOCK name="VAB_Florence_20080709_1_1000G_10" serial="1">
```

```
    <FILES>
    <FILE filename="Florence_20080709_1_1000G_10.0001.0001_0025.srf"
        filetype="srf">
    </FILE>
    </FILES>
</DATA_BLOCK>
<DATA_BLOCK name="VAB_Florence_20080709_1_1000G_10" serial="26">
    <FILES>
    <FILE filename="Florence_20080709_1_1000G_10.0002.0026_0050.srf"
        filetype="srf">
    </FILE>
    </FILES>
</DATA_BLOCK>
```

and so on.

2.2 Multiple Samples, User De-multiplexed

The XML schema now allows you to specify multiple data blocks per run each of which is assigned to a subset of the sequencing that is associated with a particular sample. In this case the submitter has de-multiplexed the sequencing run and submitted separate files. A default file may be used to contain the reads that did not get assigned to a particular sample. The **DATA_BLOCK.member** attribute records the pool member name that the reads should be assigned to.

```
<DATA_BLOCK
  serial = "1"
  name = "FMSX0OV"
  region = "1"
  member_name = "default"
>
  <FILES>
    <FILE filename="default.sff"
          filetype="sff"
          checksum_method="MD5"
          checksum="4026fc6b91ed2ffbef374a665e02802b" />
  </FILES>
</DATA_BLOCK>
<DATA_BLOCK
  serial = "2"
  name = "FMSX0OV"
  region = "1"
  member_name = "R27Cecum"
>
 <FILES>
    <FILE filename="R27Cecum.sff"
          filetype="sff"
          checksum_method="MD5"
          checksum="7f7ba170dbc6a25409a5eb6d845da88f" />
  </FILES>
</DATA_BLOCK>
```

3 FILE Descriptor Attributes

Here is a quick guide for how to use the FILE descriptor attributes for text data. More details follow in the sections below.

Expected File Forms	**Library Type (F=Frag, P=paired)**	**Number of files**	**Filetype**	**DATA_BLOCK.name**	**READ_LABEL(s) Bracket indicate array of choices for one file.**	**DATA_SERIES_LABEL(s) Bracket indicates an array of choices for one file.**	**quality_scoring_system**	**quality_encoding**	**ascii_offset**
csfasta file qual file	F	2	SOLiD_native_csfasta SOLiD_native_qual	Flowcell/slide	F3	INSDC:read, INSDC:quality	 phred	 decimal	
csfasta file qual file	P	4	SOLiD_native	Flowcell/slide	F3, R3	INSDC:read, INSDC:quality	 phred	 decimal	
qseq file	F	1	Illumina_native_qseq	Flowcell	F	[INSDC:read, INSDC:quality]	log-odds	ascii	@
fastq file with barcode	F	2	Illumina_native_fastq	Flowcell	[F, B]	[INSDC:read, INSDC:quality]	log-odds	ascii	@
qseq file	P	2	Illumina_native_qseq	Flowcell	[F, R]	[INSDC:read, INSDC:quality]	log-odds	ascii	@
fastq files with barcode	P	3	Illumina_native_fastq	Flowcell	[F,R,B]	[INSDC:read, INSDC:quality]	log-odds	ascii	@
qseq int	F	2	Illumina_native_qseq Illumina_native_int	Flowcell	F, I	[INSDC:read, INSDC:quality] INSDC:intensity	log-odds	ascii	@
qseq int	P	3	Illumina_native_qseq Illumina_native_int	Flowcell	[F, R], I	[INSDC:read, INSDC:quality] INSDC:intensity	log-odds	ascii	@
seq, prb	F/P	2	Illumina_native_seq Illumina_native_prb	Flowcell	F,R	INSDC:read, INSDC:quality	log-odds	decimal	
seq, prb, int	F/P	3	Illumina_native_seq Illumina_native_prb Illumina_native_int	Flowcell	F,R	INSDC:read, INSDC:quality INSDC:intensity	log-odds	decimal	
seq, fna qual	F/P	2	454_native_seq 454_native_qual	Plate		INSDC:read, INSDC:quality	 phred	 decimal	
fastq	F	1	Helicos_native	Flowcell		[INSDC:read, INSDC:quality]	 phred	 ascii	@
fastq with decimal quality scores	F/P	1	fastq	Not used		[INSDC:read, INSDC:quality]	phred	decimal	
Fastq with character quality scores	F/P	1	fastq	Not used		[INSDC:read, INSDC:quality]	log-odds	ascii	@ or !

Table 1 - File input use cases and DATA_BLOCK programming settings

3.1 Multiple Segments

The submitter may present different parts of the spot sequence in distinct files. The records must exist in both files and be in the same order. The **DATA_BLOCK. FILES.FILE.READ_LABEL** connects the file with the named read in a spot descriptor.

For a certain spot descriptor:

```
<SPOT_DESCRIPTOR>
    <SPOT_DECODE_SPEC>
        <NUMBER_OF_READS_PER_SPOT>2</NUMBER_OF_READS_PER_SPOT>
        <READ_SPEC>
            <READ_INDEX>1</READ_INDEX>
            <READ_LABEL>forward</READ_LABEL>
            <READ_CLASS>Application Read</READ_CLASS>
            <READ_TYPE>Forward</READ_TYPE>
            <BASE_COORD>1</BASE_COORD>
        </READ_SPEC>
        <READ_SPEC>
            <READ_INDEX>2</READ_INDEX>
            <READ_LABEL>reverse</READ_LABEL>
            <READ_CLASS>Application Read</READ_CLASS>
            <READ_TYPE>Reverse</READ_TYPE>
            <BASE_COORD>37</BASE_COORD>
        </READ_SPEC>
    </SPOT_DECODE_SPEC>
</SPOT_DESCRIPTOR>
```

can have the associated RUN code:

```
<DATA_BLOCK name = "HWX170-FC8080_1000" sector="1">
    <FILES>
        <FILE filename="HWX170-FC8080_1000_1_1_qseq.txt"
            filetype="Illumina_native_qseq"
            checksum_method="MD5"
            checksum="d41d8cd98f00b204e9800998ecf8427e">
            <READ_LABEL>F</READ_LABEL>
            <DATA_SERIES_LABEL>INSDC:read</DATA_SERIES_LABEL>
            <DATA_SERIES_LABEL>INSDC:quality</DATA_SERIES_LABEL>
        </FILE>
        <FILE filename="HWX170-FC8080_1000_1_2_qseq.txt"
            filetype="Illumina_native_qseq"
            checksum_method="MD5"
            checksum="204e9800998ecf8427ed41d8cd98f00b">
            <READ_LABEL>R</READ_LABEL>
            <DATA_SERIES_LABEL>INSDC:read</DATA_SERIES_LABEL>
            <DATA_SERIES_LABEL>INSDC:quality</DATA_SERIES_LABEL>
        </FILE>
    </FILES>
</DATA_BLOCK>
```

3.2 Multiple Data Series

A native format submission may consist of a single data block containing multiple data series (columns) each represented by a distinct file. The **DATA_BLOCK. FILES.FILE.DATA_SERIES_LABEL** can be used to define a precise mapping between components and columns.

```
<DATA_BLOCK>
     <FILES>
         <FILE filename='Solid0044_20081126_2_F3.csfasta'
             filetype="SOLiD_native_csfasta"
             checksum_method="MD5"
             checksum="d41d8cd98f00b204e9800998ecf8427e" >
             <DATA_SERIES_LABEL>INSDC:read</DATA_SERIES_LABEL>
         </FILE>

         <FILE filename='Solid0044_20081126_2_F3_QV.qual'
             filetype="SOLiD_native_qual"
             checksum_method="MD5"
             checksum="9800998ecf8427ed41d8cd98f00b204e">
             <DATA_SERIES_LABEL>INSDC:quality</DATA_SERIES_LABEL>
         </FILE>
     </FILES>
 </DATA_BLOCK>
```

3.3 Combining Segments and Data Series

The two parameters **DATA_BLOCK.FILES.FILE.READ_LABEL** and **DATA_BLOCK.FILES.FILE.DATA_SERIES_LABEL** can be combined into a two dimensional specification of files to segments and columns.

```
< DATA_BLOCK>
   < FILES>
     < FILE filename= 'Solid0044_20081126_2_F3.csfasta'
       filetype= "SOLiD_native_csfasta"
       checksum_method= "MD5"
       checksum= "d41d8cd98f00b204e9800998ecf8427e">
       < READ_LABEL> F3< /READ_LABEL>
       < DATA_SERIES_LABEL> INSDC:read< /DATA_SERIES_LABEL>
     < /FILE>
     < FILE filename= 'Solid0044_20081126_2_F3_QV.qual'
       filetype= "SOLiD_native_qual"
       checksum_method= "MD5"
       checksum= "9800998ecf8427ed41d8cd98f00b204e">
       < READ_LABEL> F3< /READ_LABEL>
       < DATA_SERIES_LABEL> INSDC:quality< /DATA_SERIES_LABEL>
     < /FILE>
     < FILE filename= 'Solid0044_20081126_2_R3.csfasta'
       filetype= "SOLiD_native_csfasta"
       checksum_method= "MD5"
       checksum= "4d1d8cd98f00b204e9800998ecf8427e" >
       < READ_LABEL> R3< /READ_LABEL>
       < DATA_SERIES_LABEL> INSDC:read< /DATA_SERIES_LABEL>
     < /FILE>
     < FILE filename= 'Solid0044_20081126_2_R3_QV.qual'
       filetype= "SOLiD_native_qual"
       checksum_method= "MD5"
       checksum= "8900998ecf8427ed41d8cd98f00b204e">
       < READ_LABEL> R3< /READ_LABEL>
       < DATA_SERIES_LABEL> INSDC:quality< /DATA_SERIES_LABEL>
     < /FILE>
   < /FILES>
< /DATA_BLOCK>
```

3.4 Specifying Qualities

Quality forms are particularly problematic as their formats are not well constrained. To better support this form of submission certain DATA_BLOCK parameters can be used to reduce the ambiguity of the input data.

The **DATA_BLOCK.FILES.FILE.quality_scoring_system** parameter can be used to specify whether the quality scores encountered in the fastq file are phred scale or log-odds scale. The SRA will convert log-odds into phred, but to do this properly the loader must know whether the log-odds scale is being used. For example:

```
< DATA_BLOCK name= "KN-930" sector= "1">
  < FILES>
    < FILE filename= "KN-930_1.fastq"
      filetype= "fastq"
      quality_scoring_system= "log-odds"
      quality_encoding= “ascii”
      ascii_offset= "@">
      < DATA_SERIES_LABEL> INSDC:read< /DATA_SERIES_LABEL>
      < DATA_SERIES_LABEL> INSDC:quality< /DATA_SERIES_LABEL>
    < /FILE>
  < /FILES>
< /DATA_BLOCK>
```

The **DATA_BLOCK.FILES.FILE.quality_encoding** parameter can tell whether the quality string in the fastq or native file is an ASCII character based string or an array of decimal values.

The **DATA_BLOCK.FILES.FILE.ascii_offset** tells which character is used as the basis (the zero) for the quality scores (choices are ascii 33 (!) or ascii 64(@)). Note that values can be negative. Negative values may be valid if the **DATA_BLOCK.FILES.FILE.quality_scoring_system** parameter is set to “log-odds”.

For example, consider the following sequencing data files :

```
gizmo2> sffinfo -s EAY20JP03.fna | head -n 2

>EAY20JP03GX7O6
GGGGGGGGGTAGGGGATGATGCCTTTGCAGTCAGTGCGGTGTCTGACAGCAACAGTGAGA

gizmo2> sffinfo -q EAY20JP03.qual | head -n 2

>EAY20JP03GX7O6
27 18 13 10 7 5 3 1 1 20 25 41 34 21 9 28 24 28 24 28 28 35 25 38 31 14 28 28 28 28 27
28 28 27 28 28 28 35 26 28 27 28 28 28 28 27 28 25 27 28 35 25 28 28 28 25 28 25 28 25
```

These can be represented with the following XML:

```
< DATA_BLOCK>
   < FILES>
      < FILE filename= "EAY20JP03.fna"
         filetype= "454_native_seq"
         checksum_method= "MD5"
         checksum= "d41d8cd98f00b204e9800998ecf8427e">
         < DATA_SERIES_LABEL> INSDC:read< /DATA_SERIES_LABEL>
      < /FILE>
      < FILE filename= "EAY20JP03.qual"
         filetype= "454_native_qual"
         checksum_method= "MD5"
         checksum= "9800998ecf8427ed41d8cd98f00b204e"
         quality_encoding= "decimal">
         < DATA_SERIES_LABEL> INSDC:quality< /DATA_SERIES_LABEL>
      < /FILE>
   < /FILES>
< /DATA_BLOCK>
```

Another example :

```
< DATA_BLOCK>
   < FILES>
      < FILE filename= "s_7_sequence.fastq"
         filetype= "fastq"
         checksum_method= "MD5"
         checksum= "d41d8cd98f00b204e9800998ecf8427e"
         quality_scoring_system= "phred"
         quality_encoding= “ascii”
         ascii_offset= "!">
         < DATA_SERIES_LABEL> INSDC:read< /DATA_SERIES_LABEL>
         < DATA_SERIES_LABEL> INSDC:quality< /DATA_SERIES_LABEL>
      < /FILE>
   < /FILES>
< /DATA_BLOCK>
```

Note that even with the offset of @, negative values (down to -5) may be generated by the decoding.

3.5 Using Filename and Checksum Attributes

New FILE attributes of checksum_method and checksum have been introduced in order to provide the loader a specification of what files to actually load. This separates concerns of verifying that a transmission of data arrived intact at NCBI, and the need to direct the loader's activities to individual components within that transmission. The combination of filename and checksum are used to verify file identity and integrity in both cases. Consequently, the **RUN.DATA_BLOCK.FILES.FILE.filename** and **RUN.DATA_BLOCK.FILES.FILE.checksum** can, but do not have to be the same values entered into the **SUBMISSION.FILES.FILE.filename** and **SUBMISSION.FILES.FILE.checksum**.

4 Implications for Loader Design

The changes in this document imply changes to both the Toolkit and the submission pipeline.

a. SRA, SRF, SFF file types are determined by their magic number (file command). If this fails, they are regarded as "Text" files. Text files are either one of the "native" platform types, or generic "fastq".

b. Filenames are NOT relied upon in order to decide how to process the content of containers and of generic fastq. The XML must specify the necessary information. This protects against the tendency for instrument manufacturers to change the file names of their standard output files. Native file formats do have constrained filenames as defined by the manufacturer.

c. The final archival representation of single dimensional quality scores is always the phred scoring system. The DATA_BLOCK settings exist in order to tell the loader how to interpret the input data. Log-odds representation is converted to phred representation as part of the loading process. The original log-odds scores are NOT preserved in the SRA.

d. "Native" loaders use grammars specified in the SRA File Formats Guide. If a loader is asked to parse files according to a certain native file model (for example, "Illumina_native", it uses a limited set of grammars to determine the filetype of each input file. If an input file fails to match one of the grammars, the load should fail. The loader should indicate on which line of input the failure occurred.

e. The "Fastq" loader uses a set of grammars specified in the SRA File Formats Guide. If no grammar matches the input file, the load fails. The loader should indicate on which line of input the failure occurred.

f. Read names are specified as spot addresses according to the naming rule for each "native" grammar. For native loaders, read names must be unique in the file, be of the same order if found split between multiple files, and be within reasonable ranges determined by the vendor. Read names of runs successfully loaded with "native" loaders are indexed. -0 is rounded to 0.

g. Read names are not interpreted, indexed, or tested for uniqueness in files with filetype "fastq". Read names from "fastq" type input are NOT preserved in the SRA. Consequently, secondary analysis depending on fastq input cannot be processed by NCBI, because of the inability to link read names before accessioning.

h. The "Fastq" filetype does NOT support mate pairs, multiply segmented reads (each of which is in a different file), or data series that are found in different files.

i. If an input file cannot be interpreted according to a solution in Table 1 - File input use cases and DATA_BLOCK programming settings then the input is rejected and the load fails. The user visible error message should indicate "Submitted filetype or format is not supported."

j. The loader should fail to load the run if any data block among several data blocks fails to load, and indicate which data block failed to load.

k. Loading may terminate with exception at the first instance of error without the need to determine further errors in that load attempt. The loader should indicate which file caused the exception.

l. The loader should be repeatable given the same input and the initial starting state. There should be a method in the loading pipeline to reinitialize a load once an exception has been thrown and loading has stopped.

m. A 454_native loader should be developed.

n. A Helicos_native loader should be developed.

o. The loaders that limit the number of data blocks that can be loaded in one run need to be changed to accommodate the serial attribute that orders the load of data blocks.

p. A new column in the SRA corresponding to member_name should be added in order to store the member assignment from a user-demultiplexed bar code run.

q. The data series types defined in this document should be added to the Toolkit:

- INSDC:read
- INSDC:read_filter
- INSDC:quality
- INSDC:intensity
- INSDC:signal
- INSDC:noise
- INSDC:position
- INSDC:clip_quality_left
- INSDC:clip_quality_right
- INSDC:tab
- INSDC:readname
- INSDC:read_seg