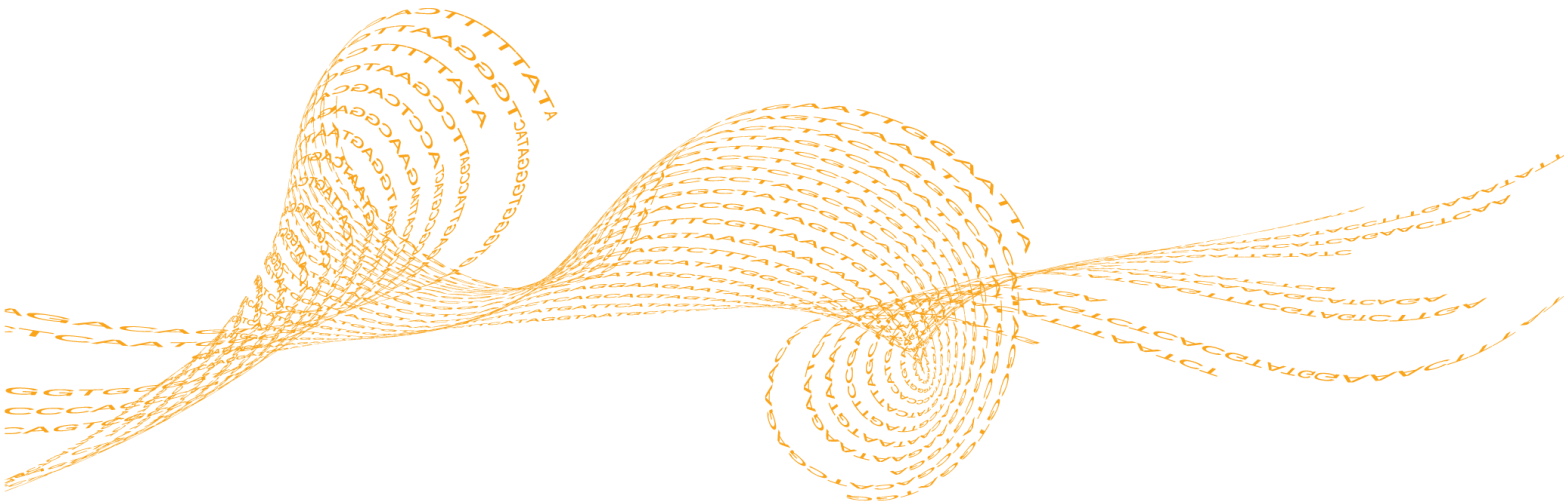
# bcl2fastq Conversion

# User Guide

Version 1.8.4

FOR RESEARCH USE ONLY






ILLUMINA PROPRIETARY

Part # 15038058 Rev B

March 2013


illumina®

# Introduction

Illumina sequencing instruments generate per-cycle BCL basecall files as primary sequencing output, but many downstream analysis applications use per-read FASTQ files as input. bcl2fastq combines these per-cycle BCL files from a run and translates them into FASTQ files. bcl2fastq can begin bcl conversion as soon as the first read has been completely sequenced.

At the same time as converting, bcl2fastq also separates multiplexed samples (demultiplexing). Multiplexed sequencing allows you to run multiple individual samples in one lane. The samples are identified by index sequences that were attached to the template during sample prep. The multiplexed sample FASTQ files are assigned to projects and samples based on a user-generated sample sheet, and stored in corresponding project and sample directories (see also *Generating the Sample Sheet* on page 13). If no sample sheet is provided, a single Project directory is created named with a suffix of the FlowCell ID. Within this Project directory is a single Sample directory for each lane (e.g. Sample_lane1).

Each directory can be independently analyzed (alignment, variant analysis, and counting) with downstream analysis software. The output of bcl2fastq is fully compatible with the alignment and variant calling components of CASAVA 1.8.

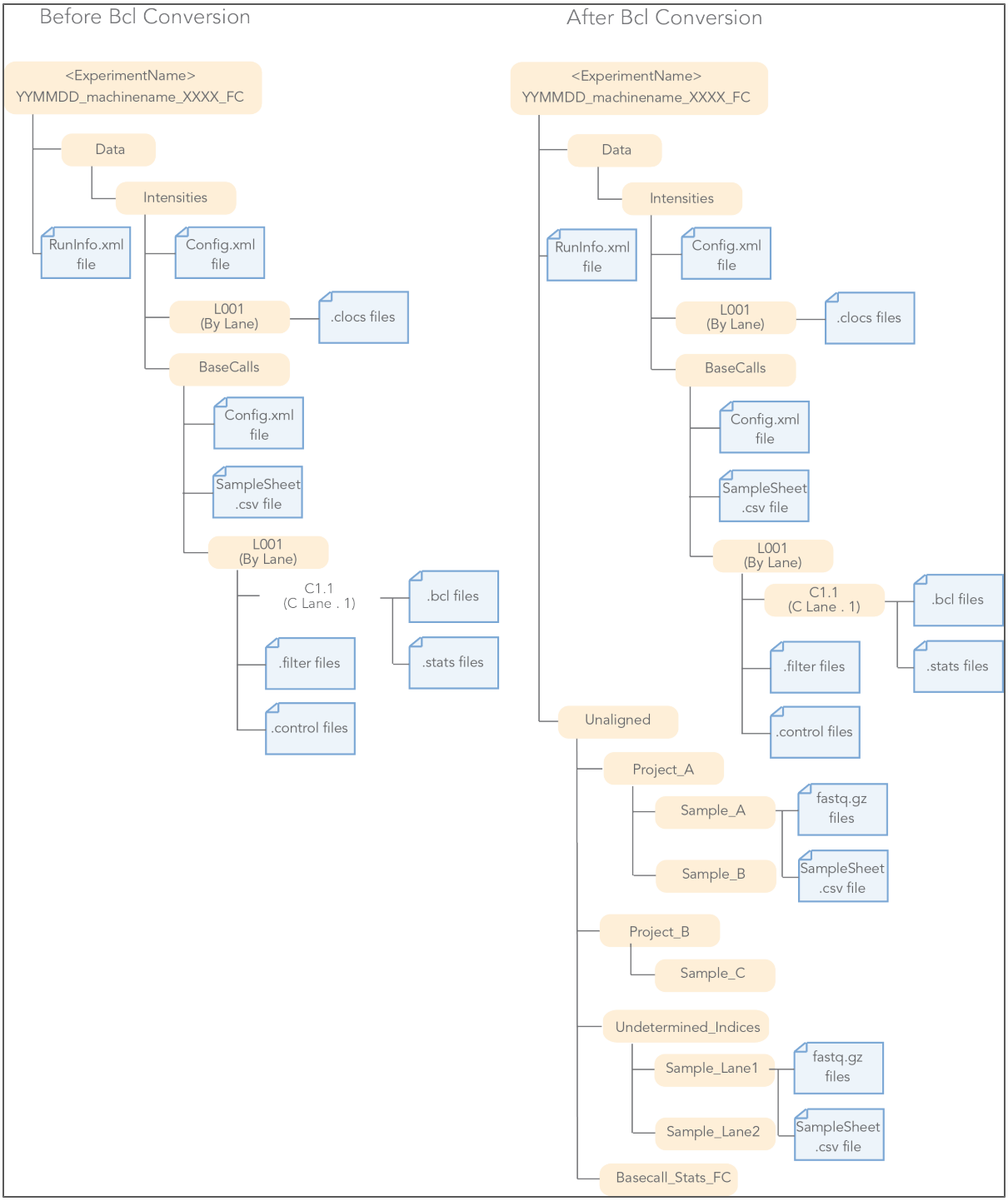
During the conversion step, adapter masking may also be performed. With this feature, bcl2fastq will check whether a read has proceeded past the genomic insert and into adapter sequence. If an adapter sequence is detected, the corresponding base calls beyond the match will be changed to N in the resultant FASTQ file.

## Bcl Conversion/Demultiplexing Directory Structure

Bcl conversion and demultiplexing is done in a single step, and generates a new directory in the Run folder (with default name **Unaligned**), which contains all of the demultiplexed compressed FASTQ files. One level down from the Unaligned directory are the **Project directories** and within each project directory are the **Sample directories**.

Reads with undetermined indices will be placed in the directory Undetermined_indices, unless the sample sheet specifies a specific sample and project for reads without index in that lane.

Figure 1 Typical Run Folder Structure after Bcl Conversion and Demultiplexing




## Sample Sheet

The sample sheet (SampleSheet.csv file) directs the software how to assign reads to samples, and samples to projects. The sample sheet specifies for every index in every lane which sample and which project it belongs to. Lanes with samples that were not indexed can also be assigned to samples and projects using the sample sheet. Projects

can consist of multiple samples, and samples can consist of multiple indices which may come from multiple lanes.

The sample sheet contains the following columns:

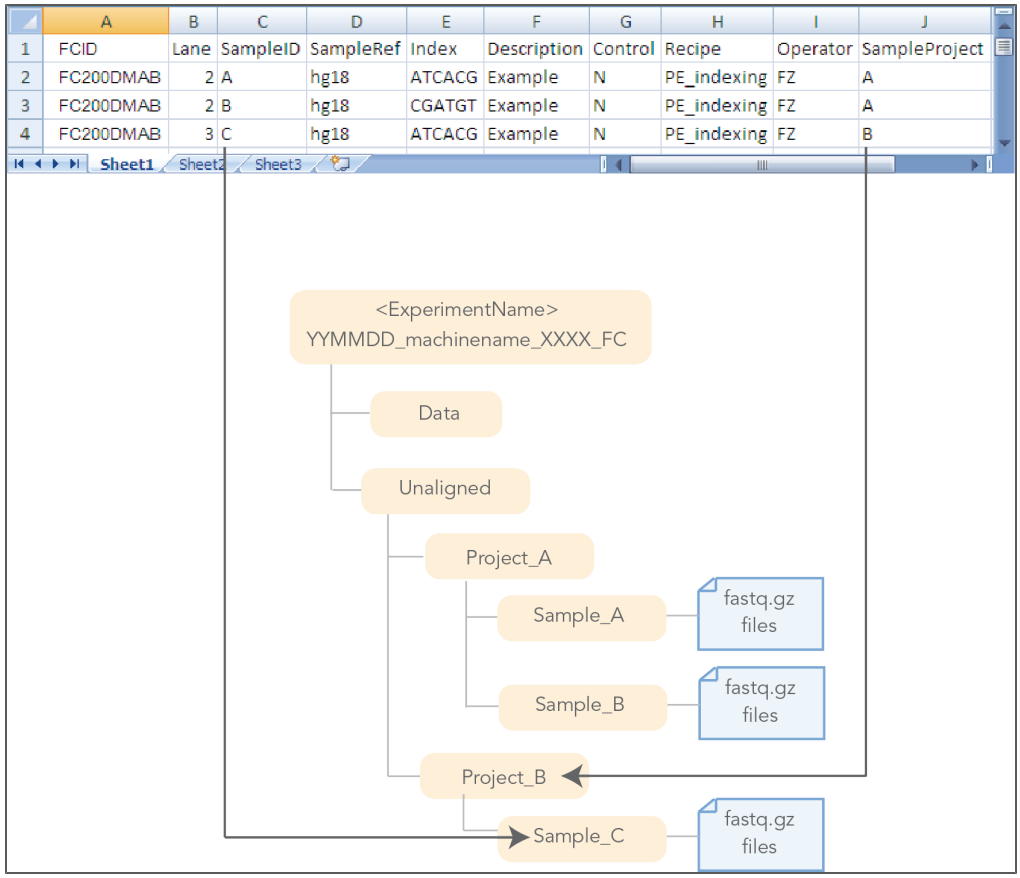
| Column | Description |
|---|---|
| FCID | Flow cell ID |
| Lane | Positive integer, indicating the lane number (1-8) |
| SampleID | ID of the sample |
| SampleRef | The name of the reference |
| Index | Index sequence(s) |
| Description | Description of the sample |
| Control | Y indicates this lane is a control lane, N means sample |
| Recipe | Recipe used during sequencing |
| Operator | Name or ID of the operator |
| SampleProject | The project the sample belongs to |

Every project in the sample sheet is linked to a corresponding project directory. Each sample belonging to that project is linked to a corresponding sample directory within that project directory. Reads are stored in the FASTQ files located in the project and sample directories specified in the sample sheet, as illustrated below for the sample in line 4 of the sample sheet.

NOTE

The Illumina Experiment Manager (IEM) helps you with your sample sheet. IEM is a wizard-driven application that guides you through the creation and setup of your sample sheet. IEM can be run on any Windows platform. You can download it from the Illumina website at http://www.illumina.com. A MyIllumina account is required.

Figure 2 Relation between Sample Sheet and Directory Structure




## Bcl Conversion/Demultiplexing Examples

Bcl conversion and demultiplexing support four scenarios:

- Multiplexed samples, with sample sheet.
  Reads are placed within the directory structure specified by the sample sheet, based on the index and lane information. Reads for which the index sequence was ambiguous will be placed in a project directory called Undetermined_indices, unless the sample sheet specifies a specific sample and project for reads without index in that lane.
- Both multiplexed and non-multiplexed samples present, with sample sheet.
  Reads are placed within the directory structure specified by the sample sheet, based on the index and lane information. Reads containing ambiguous or no barcodes will be placed in a project directory called Undetermined_indices, unless the sample sheet specifies a specific sample and project for reads without index in that lane. Non-multiplexed samples are listed with no entry in the index field.
- Samples are not multiplexed, with sample sheet.
  Reads are placed within the directory structure directed by the sample sheet, based on the lane they are originating from.
- Samples are not multiplexed, without sample sheet.
  Reads are placed in a project directory named after the flow cell, and sample directories based on the lane the reads are originating from.

## Demultiplexing Method

Demultiplexing involves reorganizing the FASTQ files based on the index information, and generating the statistics and reporting files. This section describes these two steps.

### Reorganizing FASTQ Files

The first step of demultiplexing in bcl2fastq is reorganizing the base call files, based on the index sequence. This is done the following way for each cluster:

1. Get the raw index for each index read from the .bcl file.
2. Identify the appropriate directory for the index based on the sample sheet.
3. ***Optional***: Detect and correct up to one error on the barcode, and identify the appropriate directory. If there are multiple index reads, detect and correct up to one error in each index read.
4. ***Optional***: Detect the presence of adapter sequence at the end of read. If adapter sequence is detected, mask the corresponding basecalls with N.
5. For each read:
   a. Write the index sequence into the index field.
   b. Append the read to the appropriate new FASTQ file in the selected directory.
6. If the index cannot be identified, the data is written into the Undetermined_indices directory, unless the sample sheet specifies a project and sample for reads without index.

### Updating Statistics and Reporting

The sample demultiplexer performs additional tasks:

- Generates statistics
  While splitting the FASTQ files, bcl2fastq recalculates the base calling analysis statistic that were computed during base calling for the unsplit lanes. These files are stored in the Unaligned/Basecall_Stats_FCID folder.
- Regenerates the analysis plots for each multiplexed sample
- Copies raw matrix and phasing files
- Updates sample sheet
  The sample demultiplexer strips all the non-relevant indexes from the original sample sheet and places the stripped out version in the appropriate directory.

For a description of these files, see *Bcl Conversion Output Folder* on page 20.

bcl2fastq can be downloaded from MyIllumina (https://my.illumina.com) as RPM package or tarball.

NOTE
For installation requirements, see *Appendix: Requirements and Software Installation* on page 26.

## Installing from RPM Package

Root access to the system is a pre-requisite for this installation procedure. The command line for installing the RPM file is as follows:

```
sudo rpm -i path-to-RPM-file/bcl2fastq-1.8.4-Linux-x86_64.rpm
```

The starting point for the bcl2fastq converter will be located in /usr/local/bin/configureBclToFastq.pl.

## Installing from Source

This installation procedure uses the following directory locations:

- Source and build directories are in the directory indicated by the environment variable “TMP”
- The package is installed in the directory indicated by the environment variable “INSTALL”

For example, these environment variables could be set as:

```
export TMP=/tmp
export SOURCE=${TMP}/bcl2fastq-1.8.4
export BUILD=${TMP}/bcl2fastq-1.8.4-build
export INSTALL=/usr/local/bcl2fastq-1.8.4
```

NOTE
The build directory must be different from the source directory.

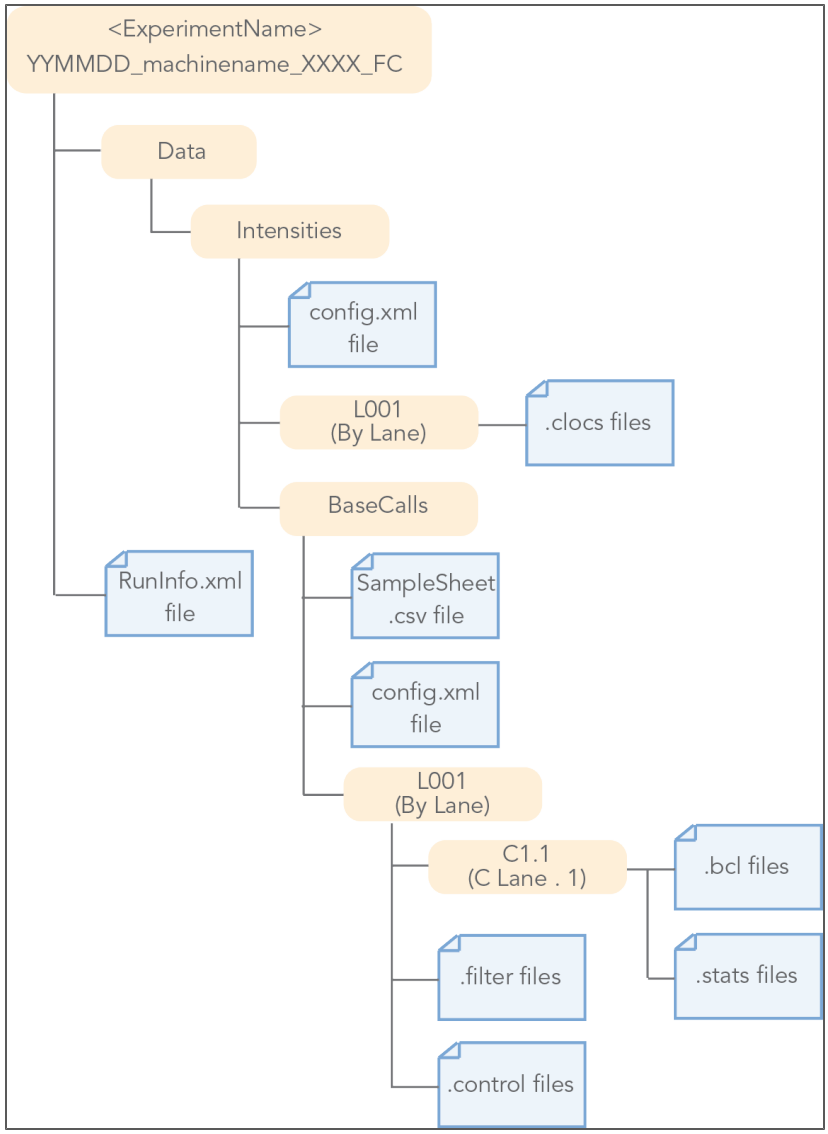
The install procedure follows the usual steps: decompressing and extracting the source, configuring the build, building the package, and installing:

1. Decompressing and extracting the source code:
   ```
   cd ${TMP}
   tar xjf path-to-tarball/bcl2fastq-1.8.4.tar.bz2
   ```
   This will create a bcl2fastq-1.8.4 sub-directory in the ${TMP} directory.
2. Configuring the build:
   ```
   mkdir ${BUILD}
   cd ${BUILD}
   ${SOURCE}/src/configure --prefix=${INSTALL}
   ```
3. Building:
   ```
   make
   ```
4. Installing (might require root privilege, depending on the ${INSTALL} path):
   ```
   make install
   ```

# Bcl Conversion Input Files

Demultiplexing needs a BaseCalls directory and a sample sheet to start a run. These files are described below.

Figure 3 Bcl Conversion Input Files




## Folder and File Naming

The top level run folder name is generated using three fields to identify the <ExperimentName>, separated by underscores. For example:

```
YYMMDD_machinename_NNNN
```

You should not deviate from the run folder naming convention, as this may cause the software to stop.

1 The first field is a six-digit number specifying the date of the run. The YYMMDD ordering ensures that a numerical sort of run folders places the names in chronological order.

2 The second field specifies the name of the sequencing machine. It may consist of any combination of upper or lower case letters, digits, or hyphens, but may ***not*** contain any other characters (especially not an underscore). It is assumed that the sequencing platform is synonymous with the PC controlling it, and that the names assigned to the instruments are unique across the sequencing facility.

3 The third field is a four-digit counter specifying the experiment ID on that instrument. Each instrument should be capable of supplying a series of consecutively numbered experiment IDs (incremental unique index) from the onboard sample tracking database or a LIMS.

NOTE
It is desirable to keep Experiment-IDs (or Sample-ID) and instrument names unique within any given enterprise. You should establish a convention under which each machine is able to allocate run folder names independently of other machines to avoid naming conflicts.

A run folder named 120108_instrument1_0147 indicates experiment number 147, run on instrument 1, on the 8th of Jan 2012. While the date and instrument name specify a unique run folder for any number of instruments, the addition of an experiment ID ensures both uniqueness and the ability to relate the contents of the run folder back to a laboratory notebook or LIMS.

Additional information is captured in the run folder name in fields separated by an underscore from the first three fields. For example, you may want to capture the flow cell number in the run folder name as follows: YYMMDD_machinename_XXXX_FCYYY.

NOTE
When publishing the data to a public database, it is desirable to extend the exclusivity globally, for instance by prefixing each machine with the identity of the sequencing center.

## BaseCalls Directory

Demultiplexing requires a BaseCalls directory containing the binary base call files (BCL files) as generated by RTA, OLB (Off-Line Basecaller), or RTAOLB.

The BCL to FASTQ converter needs the following input files from the BaseCalls directory:

- BCL files.
- *.stats files.
- *.filter files.
- *.control files
- *.clocs, *.locs, or *_pos.txt files. The BCL to FASTQ converter determines which type of position file it looks for based on the RTA version that was used to generate them.
- RunInfo.xml file. The RunInfo.xml is at the top level of the run folder.
- config.xml file

RTA is configured to copy these files off the instrument computer machine to the BaseCalls directory on the analysis server.

### BCL Files

The BCL files can be found in the BaseCalls directory inside the run directory:

`Data/Intensities/BaseCalls/L<lane>/C<cycle>.1`

They are named as follows:

`s_<lane>_<tile>.bcl` or `s_<lane>_<tile>.bcl.gz`

The BCL files can be gzip compressed and the FASTQ generator accepts either format. The BCL files are binary base call files with the format described below.

| Bytes | Description | Data type |
|---|---|---|
| Bytes 0–3 | Number N of cluster | Unsigned 32bits little endian integer |
| Bytes 4–(N+3)<br>Where N is the cluster index | Bits 0-1 are the bases, respectively [A, C, G, T] for [0, 1, 2, 3]:<br>bits 2-7 are shifted by two bits and contain the quality score.<br>All bits '0' in a byte is reserved for no-call. | Unsigned 8bits integer |

## Stats Files

The stats files can be found in the BaseCalls directory inside the run directory:

```
Data/Intensities/BaseCalls/L00<lane>/C<cycle>.1
```

They are named as follows:

```
s_<lane>_<tile>.stats
```

The Stats file is a binary file containing base calling statistics; the content is described below. The data is for clusters passing filter only.

| Start | Description | Data type |
|---|---|---|
| Byte 0 | Cycle number | integer |
| Byte 4 | Average Cycle Intensity | double |
| Byte 12 | Average intensity for A over all clusters with intensity for A | double |
| Byte 20 | Average intensity for C over all clusters with intensity for C | double |
| Byte 28 | Average intensity for G over all clusters with intensity for G | double |
| Byte 36 | Average intensity for T over all clusters with intensity for T | double |
| Byte 44 | Average intensity for A over clusters with base call A | double |
| Byte 52 | Average intensity for C over clusters with base call C | double |
| Byte 60 | Average intensity for G over clusters with base call G | double |
| Byte 68 | Average intensity for T over clusters with base call T | double |
| Byte 76 | Number of clusters with base call A | integer |
| Byte 80 | Number of clusters with base call C | integer |
| Byte 84 | Number of clusters with base call G | integer |
| Byte 88 | Number of clusters with base call T | integer |
| Byte 92 | Number of clusters with base call X | integer |
| Byte 96 | Number of clusters with intensity for A | integer |
| Byte 100 | Number of clusters with intensity for C | integer |
| Byte 104 | Number of clusters with intensity for G | integer |
| Byte 108 | Number of clusters with intensity for T | integer |

## Filter Files

The filter files can be found in the BaseCalls directory.

The *.filter files are binary files containing filter results; the format is described below.

| Bytes | Description |
|---|---|
| Bytes 0–3 | Zero value (for backwards compatibility) |
| Bytes 4–7 | Filter format version number |
| Bytes 8–11 | Number of clusters |
| Bytes 12–(N+11)<br>Where N is the cluster number | unsigned 8-bits integer:<br>• Bit 0 is pass or failed filter |

## Control Files

The control files can be found in the BaseCalls directory:

```
<run directory>/Data/Intensities/BaseCalls/L00<lane>/
```

They are named as follows:

```
s_<lane>_<tile>.control
```

The *.control files are binary files containing control results; the format is described below.

| Bytes | Description |
|---|---|
| Bytes 0–3 | Zero value (for backwards compatibility) |
| Bytes 4–7 | Format version number |
| Bytes 8–11 | Number of clusters |
| Bytes 12–(2xN+11)<br>Where N is the cluster index | The bits are used as follows:<br>• Bit 0: always empty (0)<br>• Bit 1: was the read identified as a control?<br>• Bit 2: was the match ambiguous?<br>• Bit 3: did the read match the phiX tag?<br>• Bit 4: did the read align to match the phiX tag?<br>• Bit 5: did the read match the control index sequence?<br>• Bits 6,7: reserved for future use<br>• Bits 8..15: the report key for the matched record in the controls.fasta file (specified by the REPORT_KEY metadata) |

## Position Files

The BCL to FASTQ converter can use different types of position files and will expect a type based on the version of RTA used:

- *.locs: the locs files can be found in the Intensities/L<lane> directories.
- *.clocs: the clocs files are compressed versions of locs file and can be found in the Intensities/L<lane> directories.
- *_pos.txt: the pos files can be found in the Intensities directory.
  The *_pos.txt files are text files with 2 columns and a number of rows equal to the number of clusters. The first column is the X-coordinate and the second column is the Y-coordinate. Each line has a <cr><lf> at the end.

## RunInfo.xml File

The top level Run Folder contains a RunInfo.xml file. The file RunInfo.xml (normally generated by SCS/HCS) identifies the boundaries of the reads (including index reads).

The XML tags in the RunInfo.xml file are self-explanatory.

## config.xml Files

In the Intensities folder you will find the config.xml file that records any information specific to the generation of the subfolders. This contains a tag-value list describing the

cycle-image folders used to generate each folder of intensity and sequence files.

In the BaseCalls folder there is another config.xml file containing the meta-information about the base caller runs. These files are different, though related in content.

## Adapter Sequences Files

Specifying an adapter sequence file as the argument to the `--adapter-sequence` option causes the bcl2fastq converter to recognize this sequence in the output and N-mask the adapter sequence and later cycles when writing out FASTQ files. Masking the adapter sequence avoids reporting of spurious mismatches with the reference sequence, and improves the performance (both accuracy and speed) of alignment.

The adapter sequences can be provided in one or more FASTA file by the `--adapter-sequence` option (see *Options for Bcl Conversion and Demultiplexing* on page 16). If bcl2fastq is invoked with 1 instance of the `--adapter-sequence` option, it uses that file both for read 1 and read 2 (so the same adapter masking for both reads). If there are 2 files being pointed by two instances of the `--adapter-sequence` option, the first one will be used for read 1, whereas the second one will be used for read 2.

The FASTA files for TruSeq, Nextera Mate Pair and Nextera Enrichment are located in ${INSTALL}/share/bcl2fastq-${VERSION}/adapters. Sequences for various Illumina adapters are also listed in the Illumina Adapter Sequences Letter, available at the Illumina website.

## Generating the Sample Sheet

The user generated sample sheet (SampleSheet.csv file) describes the samples and projects in each lane, including the indexes used. The sample sheet should be located in the BaseCalls directory of the run folder. Alternatively the location of the file can be specified with the `--sample-sheet` option.

### Illumina Experiment Manager

You can create, open, and edit the sample sheet in Excel, but it is easier to use the Illumina Experiment Manager (IEM) to create them.

IEM is a wizard-driven application that guides you through the creation and setup of your sample sheet. IEM can be run on any Windows platform. You can download it from the Illumina website at http://www.illumina.com. A MyIllumina account is required.

### Sample Sheet Columns

The sample sheet contains the following columns:

| Column Header | Description |
|---|---|
| FCID | Flow cell ID |
| Lane | Positive integer, indicating the lane number (1-8) |
| SampleID | ID of the sample |
| SampleRef | The reference used for alignment for the sample |
| Index | Index sequences. Multiple index reads are separated by a hyphen (for example, `ACCAGTAA-GGACATGA`). |
| Description | Description of the sample |
| Control | Y indicates this lane is a control lane, N means sample |
| Recipe | Recipe used during sequencing |
| Operator | Name or ID of the operator |
| SampleProject | The project the sample belongs to |

## Illegal Characters

Project and sample names in the sample sheet cannot contain illegal characters not allowed by some file systems. The characters not allowed are the **space character** and the following:

```
? ( ) [ ] / \ = + < > : ; " ' , * ^ | & .
```

## Multiple Index Reads

If multiple index reads were used, each sample must be associated with an index sequence for each index read. All index sequences are specified in the **Index** field. The individual index read sequences are separated with a hyphen character (-). For example, if a particular sample was associated with the sequence `ACCAGTAA` in the first index read, and the sequence `GGACATGA` in the second index read, the index entry would be `ACCAGTAA-GGACATGA`.

## Samples Without Index

It is possible to assign samples without index to projects, sampleIDs, or other identifiers by leaving the Index field empty.

# Running Bcl Conversion and Demultiplexing

Bcl conversion and demultiplexing is configured by one script, configureBclToFastq.pl. This section describes how to perform Bcl conversion and demultiplexing in bcl2fastq.

## Usage of configureBclToFastq.pl

The standard way to run bcl conversion and demultiplexing is to first create the necessary Makefiles, which configure the run. Then you run `make` on the generated files, which executes the calculations.

1 Enter the following command to create a makefile for demultiplexing:
`/usr/local/bin/configureBclToFastq.pl[options]`

NOTE
We recommend disabling EAMSS through the use of the --no-eamss option, particularly when bcl conversion output needs to match that from other Illumina fastq-generating processes, such as MiSeq Reporter or BaseSpace fastq generation. For more information, see *--no-eamss* on page 17.

2 Move into the newly created Unaligned folder (by default) or specified by `--output-dir`.

3 Type the "make" command. Suggestions for "make" usage, depending on your workflow, are listed below.

| Make Usage | Workflow |
|---|---|
| `nohup make -j N` | Bcl conversion and demultiplexing (default). |
| `nohup make -j N r1` | Bcl conversion and demultiplexing for read 1. |

- The -j option specifies the extent of parallelization, with the options depending on the setup of your computer or computing cluster.
- The Unix nohup command redirects the standard output and error and keeps the "make" process running even if your terminal is interrupted or if you log out.

See *Makefile Options for Bcl Conversion and Demultiplexing* on page 18 for explanation of the options.

4 After the analysis is done, review the analysis for each sample.
See *Demultiplex_Stats File* on page 24.

## Example Bcl Conversion and Demultiplexing

An example of a demultiplexing run is as follows:

1 Enter:
```
/usr/local/bin/configureBclToFastq.pl --input-dir <BaseCalls_
   dir> --output-dir <Unaligned> --sample-sheet <BaseCalls_
   dir>/SampleSheet.csv --no-eamss
```

2 Go to the <Unaligned> folder:
```
cd Unaligned
```

3 Run:
```
nohup make -j 8
```

Step one will produce a set of directories in the Unaligned directory. Reads with an unresolved or erroneous index are placed in the Undetermined_indices directory.

## Options for Bcl Conversion and Demultiplexing

The options for demultiplexing are described below.

| Option | Description | Examples |
|---|---|---|
| `--fastq-cluster-count` | Maximum number of clusters per output FASTQ file. Do not go over 16000000, since this is the maximum number of reads we recommend for one ELAND process. Specify 0 to ensure creation of a single FASTQ file.<br>Defaults to 4000000. | `--fastq-cluster-count 6000000` |
| `-i, --input-dir` | Path to a BaseCalls directory.<br>Defaults to current dir | `--input-dir <BaseCalls_dir>` |
| `-o, --output-dir` | Path to demultiplexed output.<br>Defaults to <run_folder>/Unaligned<br>Note that there can be only one Unaligned directory by default. If you want an alternate unaligned directory, you have to use this option to generate a different output directory. | `--output-dir <run_folder>/Unaligned` |
| `--positions-dir` | Path to a directory containing positions files.<br>Defaults depends on the RTA version that is detected. | `--positions-dir <positions_dir>` |
| `--positions-format` | Format of the input cluster positions information. Options:<br>• .locs<br>• .clocs<br>• _pos.txt<br>Defaults to .clocs. | `--positions-format .locs` |
| `--filter-dir` | Path to a directory containing filter files.<br>Defaults depends on RTA version that is detected. | `--filter-dir <filter_dir>` |
| `--intensities-dir` | Path to a valid Intensities directory.<br>Defaults to parent of base_calls_dir. | `--intensities-dir <intensities_dir>` |
| `-s,--sample-sheet` | Path to sample sheet file.<br>Defaults to <input_dir>/SampleSheet.csv | `--sample-sheet <input_dir>/SampleSheet.csv` |
| `--tiles` | --tiles allows processing of a subset of tiles. The option takes a comma-separated list of regular expressions to match against the expected "s_<lane>_<tile>" pattern, where <lane> is the lane number (1-8) and <tile> is the 4 digit tile number (left-padded with 0s). | `--tiles=s_[2468]_[0-9][0-9][02468]5,s_1_0001` |

| Option | Description | Examples |
|---|---|---|
| `--use-bases-mask` | The `--use-bases-mask` string specifies how to use each cycle.<br>• An "n" means ignore the cycle.<br>• A "Y" (or "y") means use the cycle to generate FASTQ.<br>• An "I" means use the cycle for the index read.<br>• A number means that the previous character is repeated that many times.<br>• The read masks are separated by commas ","<br>The format for dual indexing is as follows:`--use-bases-mask Y*,I*,I*,Y*` or variations thereof as specified above.<br>If this option is not specified, the mask will be determined from the 'RunInfo.xml' file in the run directory. If it cannot do this, you will have to supply the `--use-bases-mask`. | `--use-bases-mask y50n,I6n,Y50n`<br>`This means:`<br>• Use first 50 bases for first read (Y50)<br>• Ignore the next (n)<br>• Use 6 bases for index (I6)<br>• Ignore next (n)<br>• Use 50 bases for second read (Y50)<br>• Ignore next (n) |
| `--no-eamss` | Disable the masking of the quality values with the Read Segment Quality control metric filter.<br>The EAMMS algorithm, or Read Segment Quality Control Metric, identifies segments at the end of reads that have low quality and may have unreliable quality scores. EAMSS replaces quality scores for all basecalls in these segments with a Qscore of 2. We recommend disabling EAMSS through the use of this option, particularly when bcl conversion output needs to match that from other Illumina fastq-generating processes, such as MiSeq Reporter or BaseSpace fastq generation. EAMMS is no longer required with current Illumina sequencing technology and is not applied in such newer applications. | `--no-eamss` |
| `--mismatches` | Comma-delimited list of number of mismatches allowed for each read (for example: 1,1). If a single value is provide, all index reads will allow the same number mismatches.<br>Default is 0. | `--mismatches 1` |
| `--flowcell-id` | Use the specified string as the flowcell id. (default value is parsed from the config-file) | `--flowcell-id flow_cell_id` |
| `--ignore-missing-stats` | Fill in with zeros when *.stats files are missing | `--ignore-missing-stats` |
| `--ignore-missing-bcl` | Interpret missing *.bcl files as no call (N) | `--ignore-missing-bcl` |
| `--ignore-missing-control` | Interpret missing control files as not-set control bits | `--ignore-missing-control` |
| `--with-failed-reads` | Include failed reads into the FASTQ files (by default, only reads passing filter are included). | `--with-failed-reads` |

| Option | Description | Examples |
|---|---|---|
| `--adapter-sequence` | Path to one or more adapter sequence (FASTA) files. Each (per-read) file can have one or more contigs, which are the independent adapters to be assessed for trimming.<br>Default: None (no masking) | `--adapter-sequence <adapter_dir>/adapter.fa`<br><br>For two files, one for each read:<br>`--adapter-sequence <adapter_dir>/read1.fa --adapter-sequence <adapter_dir>/read2.fa` |
| `--adapter-stringency` | The minimum match rate that would trigger the masking process. This is calculated as `MatchCount / (MatchCount + MismatchCount)` and ranges from 0 to 1, but it is not recommended to use any value <0.5, as this would introduce too many false positives.<br>The default value for this parameter is 0.9, meaning that only reads with >90% sequence identity with the adapter will be trimmed. Should you want to replicate the behaviour from previous versions (including CASAVA), this value needs to be set at 0.667. The default behaviour replicates that of MiSeq Reporter.<br>Default: 0.9 | `--adapter-stringency 0.667` |
| `--man` | Dsiplay a manual page for this command | `--man` |
| `-h, --help` | Produce help message and exit | `-h` |

## Makefile Options for Bcl Conversion and Demultiplexing

The options for `make` usage in demultiplexing/analysis are described below.

| Parameter | Description |
|---|---|
| nohup | Use the Unix nohup command to redirect the standard output and keep the "make" process running even if your terminal is interrupted or if you log out. The standard output will be saved in a nohup.out file and stored in the location where you are executing the makefile.<br>nohup make -j n &<br>The optional "&" tells the system to run the analysis in the background, leaving you free to enter more commands.<br>We suggest always running nohup to help troubleshooting if issues arise. |
| -j N | The -j option specifies the extent of parallelization, with the options depending on the setup of your computer or computing cluster. |
| r1 | Runs Bcl conversion for read 1. Can be started once the last read has started sequencing. |
| POST_RUN_ COMMAND_R1 | A Makefile variable that can be specified either on the make command line or as an environment variable to specify the post-run commands after completion of read one, if needed. Typical use would be triggering the alignment of read 1. |
| POST_RUN_ COMMAND | A Makefile variable that can be specified on the make command line to specify the post-run commands after completion of the run. |
| KEEP_ INTERMEDIARY | The option KEEP_INTERMEDIARY tells the software not to delete the intermediary files in the Temp dir after Bcl conversion is complete. Usage: `KEEP_INTERMEDIARY:=yes` |

NOTE
If you specify one of the more specific workflows and then run a more general one, only the difference will get processed. For instance:

```
make -j N r1
```

followed by:

```
make -j N
```

will do read 1 in the first step, and read 2 the second one.

## Starting Bcl Conversion for Read 1

If you want to start Bcl to FASTQ conversion before completion of the run, use the makefile target `r1` at any time after the last read has started (for multiplexed runs, this is after completion of the indexing read).

1. Enter the following command to create a makefile for Bcl conversion:
   `/path-to-bcl2fastq/bin/configureBclToFastq.pl [options]`
2. Move into the newly created Unaligned folder specified by `--output-dir`.
3. Type the "make r1" command:
   `make -j 8 r1`

   NOTE
   the `-j <n>` command line option is supported to indicate up to <n> processes in parallel. We usually recommend no more than 16 due to IO limitations.

### Starting the Second Read

To start Bcl conversion of the second read, use the regular make command in the Unaligned folder. Perform the following:

1. Move into the Unaligned folder specified by `--output-dir`.
2. Type the regular "make" command:
   `make -j 8`
3. After the analysis is done, review the analysis for each sample.
   See *Demultiplex_Stats File* on page 24

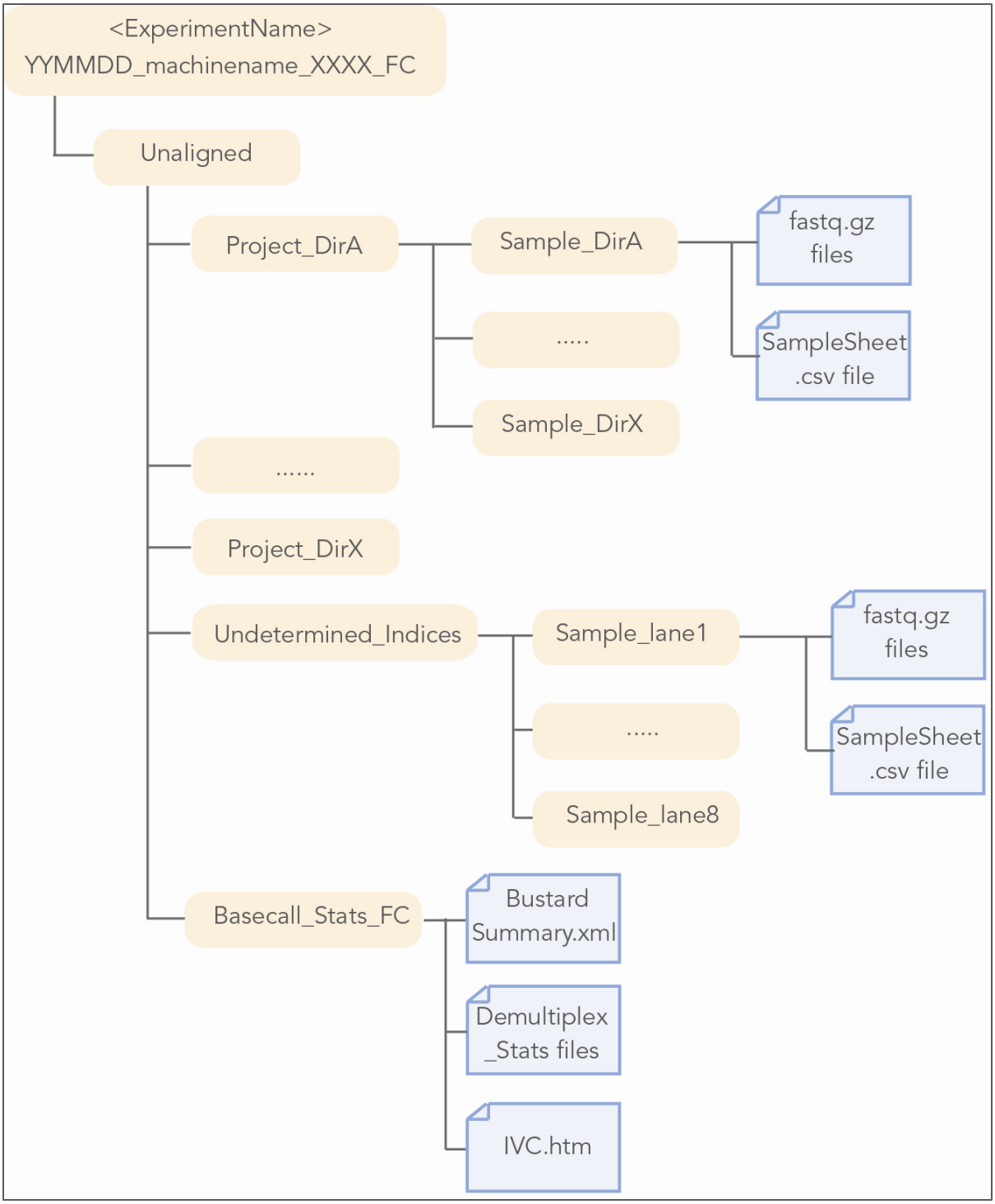
# Bcl Conversion Output Folder

The Bcl Conversion output directory has the following characteristics:

- The project and sample directory names are derived from the sample sheet.
- The Unaligned/Basecall_Stats_FCID/Demultiplex_Stats.htm file shows where the sample data are saved in the directory structure.
- The Undetermined_indices directory contains the reads with an unresolved or erroneous index.
- If no sample sheet exists, the software generates a project directory named after the flow cell, and sample directories for each lane.

NOTE

If the majority of reads end up in the 'Undetermined_indices' folder, check the --use-bases-mask parameter syntax and the length of the index in the sample sheet. It may be that you need to set the --use-bases-mask option to the length of the index in the sample sheet + the character 'n' to account for phasing.




## FASTQ Files

bcl2fastq converts *.bcl files into FASTQ files, which can be used as sequence input for alignment. The files are located in the Unaligned/Project_<ProjectName>/Sample_<SampleName> directories.

NOTE
Reads that were identified as sample prep controls in the control files are flagged as such in the FASTQ files.

## Naming

Illumina FASTQ files use the following naming scheme:

```
<sample name>_<barcode sequence>_L<lane>_R<read number>_<set
   number>.fastq.gz
```

For example, the following is a valid FASTQ file name:

```
NA10831_ATCACG_L002_R1_001.fastq.gz
```

Note that lane and set numbers are 0-padded to 3 digits.

In the case of non-multiplexed runs, <sample name> will be replaced with the lane numbers (lane1, lane2, ..., lane8) and <barcode sequence> will be replaced with "NoIndex".

## Set Size

The FASTQ files are divided in files with the file size set by the --fastq-cluster-count command line option of configureBclToFastq.pl.The different files are distinguished by the 0-padded 3-digit set number.

TIP

If you need to generate one unique fastq gzipped file for use in a third-party tool, you can set the `--fastq-cluster-count` option to 0

## Compression

FASTQ files are saved compressed in the GNU zip format, an open source file compression program. This is indicated by the .gz file extension.

## Format

Each entry in a FASTQ file consists of four lines:

- Sequence identifier
- Sequence
- Quality score identifier line (consisting of a +)
- Quality score

Each sequence identifier, the line that precedes the sequence and describes it, is in the following format:

```
@<instrument>:<run number>:<flowcell ID>:<lane>:<tile>:<x-
   pos>:<y-pos> <read>:<is filtered>:<control number>:<barcode
   sequence>
```

The elements are described below.

| Element | Requirements | Description |
|---|---|---|
| `@` | @ | Each sequence identifier line starts with @ |
| `<instrument>` | Characters allowed:<br>a-z, A-Z, 0-9 and underscore | Instrument ID |
| `<run number>` | Numerical | Run number on instrument |
| `<flowcell ID>` | Characters allowed:<br>a-z, A-Z, 0-9 | |
| `<lane>` | Numerical | Lane number |
| `<tile>` | Numerical | Tile number |

| Element | Requirements | Description |
|---|---|---|
| <x_pos> | Numerical | X coordinate of cluster |
| <y_pos> | Numerical | Y coordinate of cluster |
| <read> | Numerical | Read number. 1 can be single read or read 2 of paired-end |
| <is filtered> | Y or N | Y if the read is filtered (did not pass), N otherwise |
| <control number> | Numerical | 0 when none of the control bits are on, otherwise it is an even number. See below. |
| <barcode sequence> | ACTGCA | Barcode sequence |

An example of a valid entry is as follows; note the space preceding the read number element:

```
@EAS139:136:FC706VJ:2:5:1000:12850 1:Y:18:ATCACG
ATTCCCTATGCTAGGCTTACGATCTAGCTATCGTAC
+
BBBBCCCC?<A?BC?7@@???????DBBA@@@@A@@
```

NOTE

bcl2fastq FASTQ files contain only reads that passed filtering. If you want all reads in a FASTQ file, use the `--with-failed-reads` option.

### Control Values

The tenth columns (`<control number>`) is zero if the read is not identified as a control. If the read is identified as a control, the number is greater than zero, and the value specifies what kind of control it is. The value is the decimal representation of a bit-wise encoding scheme, with bit 0 having a decimal value of 1, bit 1 a value of 2, bit 2 a value of 4, and so on.

The bits are used as follows:

- Bit 0: always empty (0)
- Bit 1: was the read identified as a control?
- Bit 2: was the match ambiguous?
- Bit 3: did the read match the phiX tag?
- Bit 4: did the read align to match the phiX tag?
- Bit 5: did the read match the control index sequence?
- Bits 6,7: reserved for future use
- Bits 8..15: the report key for the matched record in the controls.fasta file (specified by the REPORT_KEY metadata)

## Quality Scores

A quality score (or Q-score) expresses an error probability. In particular, it serves as a convenient and compact way to communicate very small error probabilities.

Given an assertion, A, the probability that A is not true, P(~A), is expressed by a quality score, Q(A), according to the relationship:

$$Q(A) = -10 \log_{10}(P(\sim A))$$

where *P(~A)* is the estimated probability of an assertion A being wrong.

The relationship between the quality score and error probability is demonstrated with the following table:

| Quality score, Q (A) | Error probability, P (~A) |
|---|---|
| 10 | 0.1 |
| 20 | 0.01 |
| 30 | 0.001 |

### Quality Scores Encoding

Quality scores are encoded into a compact form in FASTQ files which uses only one byte per quality value. In this encoding the quality score is represented as the character with an ASCII code equal to its value + 33 (as of CASAVA v1.8). The following table demonstrates the relationship between the encoding character, the character's ASCII code, and the quality score represented.

Table 2
Table 1 ASCII Characters Encoding Q-scores 0–40

| Symbol | ASCII Code | Q-Score | Symbol | ASCII Code | Q-Score | Symbol | ASCII Code | Q-Score |
|---|---|---|---|---|---|---|---|---|
| ! | 33 | 0 | / | 47 | 14 | = | 61 | 28 |
| " | 34 | 1 | 0 | 48 | 15 | > | 62 | 29 |
| # | 35 | 2 | 1 | 49 | 16 | ? | 63 | 30 |
| $ | 36 | 3 | 2 | 50 | 17 | @ | 64 | 31 |
| % | 37 | 4 | 3 | 51 | 18 | A | 65 | 32 |
| & | 38 | 5 | 4 | 52 | 19 | B | 66 | 33 |
| ' | 39 | 6 | 5 | 53 | 20 | C | 67 | 34 |
| ( | 40 | 7 | 6 | 54 | 21 | D | 68 | 35 |
| ) | 41 | 8 | 7 | 55 | 22 | E | 69 | 36 |
| * | 42 | 9 | 8 | 56 | 23 | F | 70 | 37 |
| + | 43 | 10 | 9 | 57 | 24 | G | 71 | 38 |
| , | 44 | 11 | : | 58 | 25 | H | 72 | 39 |
| - | 45 | 12 | ; | 59 | 26 | I | 73 | 40 |
| . | 46 | 13 | < | 60 | 27 | | | |

## Demultiplex_Stats File

The Demultiplex_Stats.htm file provides stats about demultiplexing and shows where samples are saved in the directory structure. The Demultiplex_Stats file is located in the Unaligned/Basecall_Stats_FCID directory.

The file contains the sample information from the sample sheet, with added rows for reads that end up in the Undetermined_indices directory. If no sample sheet exists, bcl2fastq generates rows for each lane. The Demultiplex_Stats file has a number of additional columns that display demultiplexing stats and show the directory the samples are saved in. The Demultiplex_Stats file contains the following fields:

| Field | Description |
|---|---|
| Lane | Positive integer, indicating the lane number (1-8) |
| SampleID | ID of the sample |
| SampleRef | The reference sequence for the sample |
| Index | Index sequence |
| Description | Description of the sample |
| Control | Y indicates this lane is a control lane, N means sample |
| Project | The project the sample belongs to |
| # Reads | Number of reads, equals (total number of lines in fastq files)/4 |

| Field | Description |
|---|---|
| Yield | The sum of all bases in clusters that passed filtering for the entire project. |
| % PF | The percentage of clusters that passed filtering. |
| % of Lane | Percentage of reads in the sample compared to total number of reads in that lane. |
| % Perfect Index Reads | Percentage of index reads in this sample which perfectly matched the given index. |
| % One Mismatch Reads (Index) | Percentage of index reads in this sample which had 1 mismatch to given index. |
| % of >=Q30 Bases | Yield of bases with Q30 or higher from clusters passing filter divided by total yield of clusters passing filter. |
| Mean Quality Score | The total sum of quality scores of bases from clusters passing filter divided by total yield of bases from bases from clusters passing filter. |
| Recipe | Recipe used during sequencing |
| Operator | Name or ID of the operator |
| Directory | Full path to the directory. |

Below the sample information are links to the IVC plots.

## Finding Demultiplexed Samples

The key to finding the location of demultiplexed data is looking at the Demultiplex_Stats.htm file in the BaseCalls_Stats directory. The Directory column will indicate the project/sample output directory. The FASTQ files within the directory contain the index and lane as part of the name. Alternatively it can be inferred from the project name and the sample id as described in *FASTQ Files* on page 21.

# Appendix: Requirements and Software Installation

## Network Infrastructure

The large data volumes generated and moved when running bcl2fastq mean that you will need a high-throughput ethernet connection (at least 1 Gigabit recommended) or other data transfer mechanism.

### Storage Configurations

You can configure your analysis server with either local storage or external network storage.

- Local server storage can be internal to the server, or Direct Attached Storage (DAS), which is a separate chassis attached to the server.
  - **Internal**—Simple but not scalable. Results data must be moved off to network storage at some point to make room for subsequent runs.
  - **DAS**—External chassis that is scalable since more than one DAS can be connected to the server. The server is an application server running bcl2fastq and a file server providing access to results and receiving incoming raw data files.
- External network storage is either Network Attached Storage (NAS) or Storage Area Network (SAN). NAS and SAN are functionally equivalent, but SAN is larger, with higher performance, more connections, and more management options.
  - **NAS**—External chassis connected via an Ethernet to the server, instrument PC, and other clients on the network. NAS devices are scalable and highly optimized.
  - **SAN**—The most scalable with the highest performance. They have a very high bandwidth and support many simultaneous clients, but are complex to manage and significantly more expensive.

### Server Configurations

You can use either a single multi-processor, multi-core computer running Linux, or a cluster of Linux servers with a head node. bcl2fastq can take advantage of clustered and multi-processing servers.

- **Single multi-processor, multi-core server**—Simple but not scalable.
- **Linux Cluster**—Highly scalable and capable of running multiple jobs simultaneously. It requires one server as a management node and a minimum number of computational notes to be as efficient as a standalone server. By adding computational nodes, the cluster can service more instruments.

NOTE
We test our software with SGE; other cluster configurations (like LSF or PBS) are not recommended.

## Analysis Computer

Illumina supports running bcl2fastq only on Linux operating systems. It may be possible to run bcl2fastq on other 64-bit Unix variants, if all of the prerequisites described in this section are met.

Illumina recommends the IlluminaCompute data processing solution for bcl2fastq. IlluminaCompute is available as a multi-tier option, with the volume of instrument data

output per week determining the recommended Tier level. For more information, contact Illumina Support.

For example, for a laboratory generating 200 GB of sequence per week, the Tier 1 IlluminaCompute solution is recommended, for which the specifications are listed below (running bcl2fastq on non-IlluminaCompute systems satisfying these requirements is also supported):

- 1 APC Netshelter: 40U Rack with 1U KMM console
- 3 Dell R610 Server: 8 CPU cores, 48 GB RAM
- 3 Isilon IQ12000x storage modules
- 1 Serial MGT Console 16
- 2 Cisco 3750e switches

bcl2fastq parallelization is built around the multi-processor facilities of the "make" utility and scales very well to beyond eight nodes. Given the intensive I/O nature of Bcl conversion we recommend caution if parallelising this process beyond 16 CPU cores.

## Memory Requirements

bcl2fastq requires a minimum of 2 GB RAM per core.

## Software Requirements

bcl2fastq has been primarily developed and tested on CentOs 5, Illumina's recommended and supported platform. It *may* be possible to install and run bcl2fastq on other 64-bit Linux distributions (particularly on similar distributions such as RedHat and Fedora) or on other Unix variants, if all of the prerequisites described in this section are met.

The following software is required to run bcl2fastq; check whether it has been installed:

- GNU make (3.81 recommended)
- Perl (>= 5.8)
- libxslt
- libxslt-devel
- libxml2
- libxml2-devel
- gcc (4.0.0 or newer, except 4.0.2), with c++
- ImageMagick
- bzip2
- bzip2-devel
- zlib
- zlib-devel

Notes

# Technical Assistance

For technical assistance, contact Illumina Technical Support.

Table 3 Illumina General Contact Information

| | |
|---|---|
| **Illumina Website** | www.illumina.com |
| **Email** | techsupport@illumina.com |

Table 4 Illumina Customer Support Telephone Numbers

| Region | Contact Number | Region | Contact Number |
|---|---|---|---|
| North America | 1.800.809.4566 | Italy | 800.874909 |
| Austria | 0800.296575 | Netherlands | 0800.0223859 |
| Belgium | 0800.81102 | Norway | 800.16836 |
| Denmark | 80882346 | Spain | 900.812168 |
| Finland | 0800.918363 | Sweden | 020790181 |
| France | 0800.911850 | Switzerland | 0800.563118 |
| Germany | 0800.180.8994 | United Kingdom | 0800.917.0041 |
| Ireland | 1.800.812949 | Other countries | +44.1799.534000 |

## MSDSs

Material safety data sheets (MSDSs) are available on the Illumina website at www.illumina.com/msds.

## Product Documentation

Additional product documentation in PDF is available for download from the Illumina website. Go to www.illumina.com/support, select a product, then click **Documentation & Literature**.

Illumina
San Diego, California 92122 U.S.A.
+1.800.809.ILMN (4566)
+1.858.202.4566 (outside North America)
techsupport@illumina.com
www.illumina.com