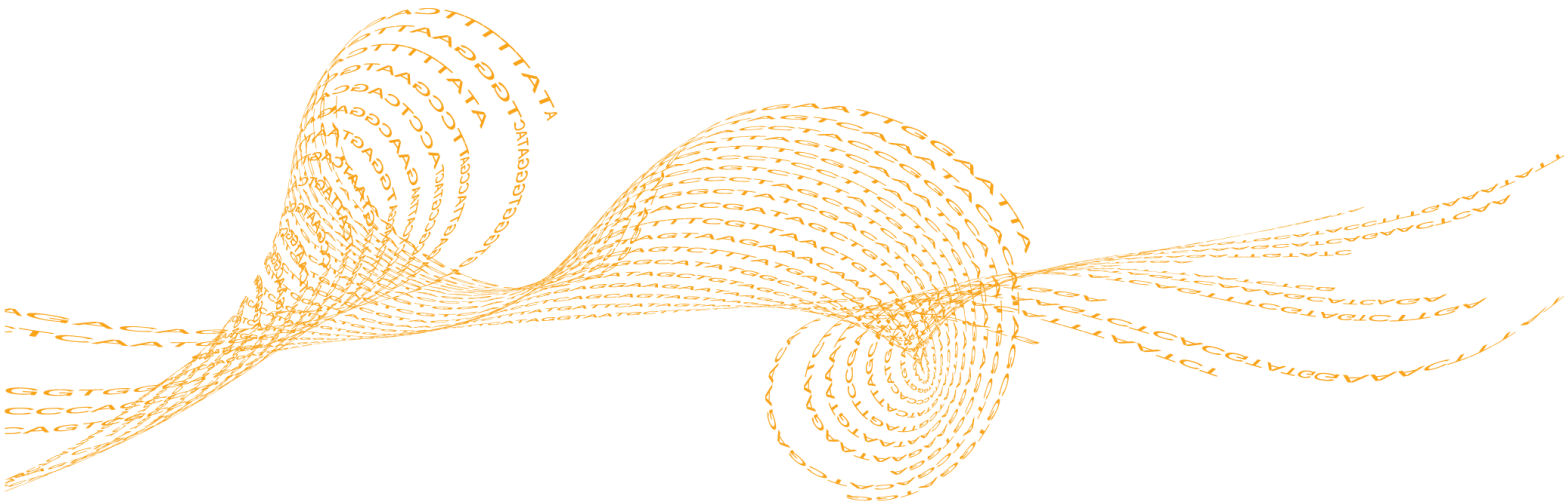
bcl2fastq Conversion

User Guide

Version v2.0

FOR RESEARCH USE ONLY



ILLUMINA PROPRIETARY

Part # 15051736 Rev A

January 2014

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.

NOTE
This documentation supports bcl2fastq version 2.0, intended for use with the NextSeq 500 only.

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 reads are assigned to samples based on a user-generated sample sheet, and stored in corresponding FASTQ files (see also *Generating the Sample Sheet* on page 11). If no sample sheet is provided, all reads are saved in Undetermined_S0_ FASTQ files.

During the conversion step, you can also perform adapter masking and trimming. With this feature, bcl2fastq checks whether a read has proceeded past the sample DNA insert and into adapter sequence. If an adapter sequence is detected, the corresponding base calls beyond the match are changed to N in the resultant FASTQ file, or removed with adapter trimming.

Supported Version

This documentation supports bcl2fastq version 2.0. This version is intended for use with the NextSeq 500.

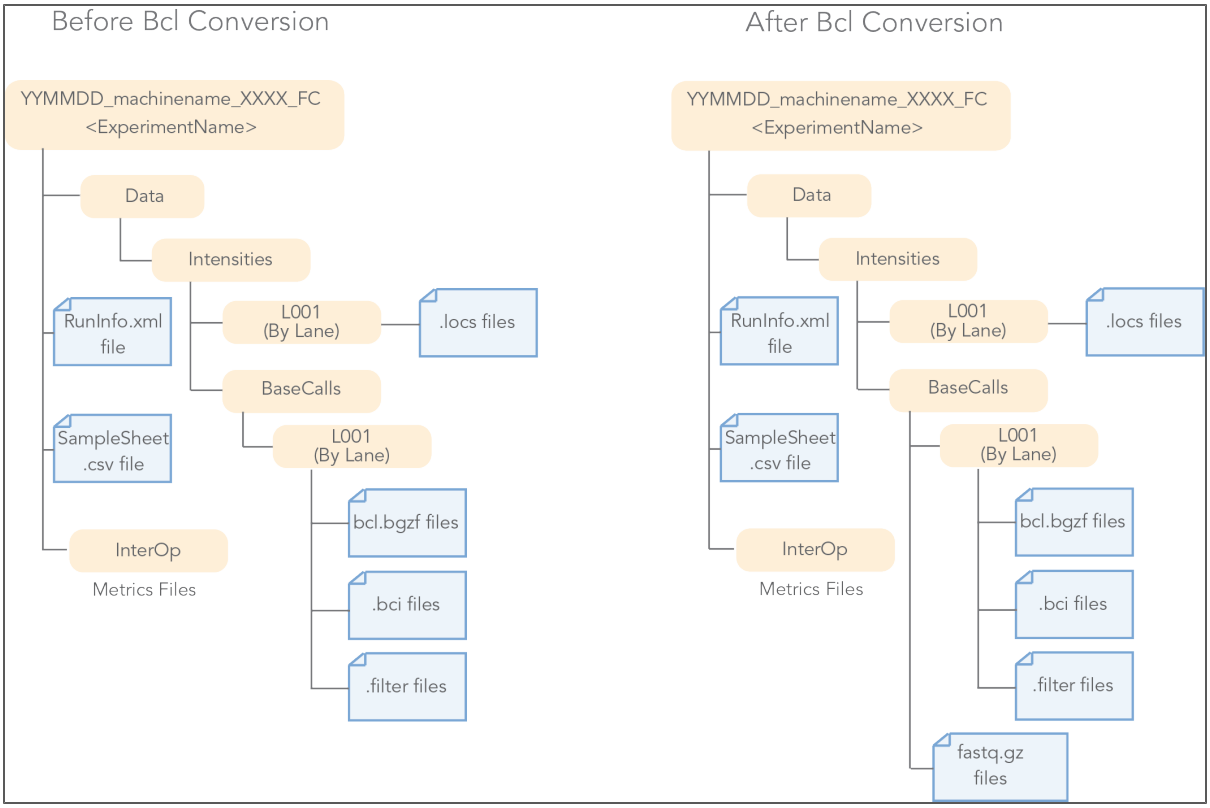
If you have a different sequencing system and want to use bcl2fastq, install bcl2fastq v1.8.4, and refer to the *bcl2fastq Conversion User Guide Version v1.8.4* for instructions.

Bcl Conversion/Demultiplexing Directory Structure

bcl2fastq performs Bcl conversion and demultiplexing in a single step, and puts the resulting demultiplexed compressed FASTQ files in <run folder>/Data/Intensities/BaseCalls.

bcl2fastq places reads with undetermined indices in files whose names start with Undetermined_S0_, unless the sample sheet specifies a specific sample for reads without index.

Figure 1 Typical Run Folder Structure after Bcl Conversion and Demultiplexing



Demultiplexing Method

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

Reorganizing FASTQ Files

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

1 Get the raw index for each index read from the *.bcl.bgzf file.

2 Identify the appropriate sample for the index based on the sample sheet.

3 ***Optional***: Detect and correct up to two errors on the barcode, and identify the appropriate sample. If there are multiple index reads, detect and correct up to two errors in each index read.

4 ***Optional***: Detect the presence of adapter sequence at the end of read. If adapter sequence is detected, mask (with N) or trim the corresponding base calls.

5 For each read:
   a Write the index sequence into the index field.
   b Append the read to the appropriate new FASTQ file.

6 If the index cannot be identified, the data is written into an Undetermined sample file, unless the sample sheet specifies a sample for reads without index.

Updating Statistics and Reporting

The sample demultiplexer also 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 InterOp folder.

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

NOTE
For installation requirements, see *Appendix: Requirements* on page 20.

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:

```
yum install -y <rpm package-name>
```

The starting point for the bcl2fastq converter is the binary executable `/usr/local/bin/bcl2fastq`.

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=${TMP}/bcl2fastq-2.0.x
export BUILD=${TMP}/bcl2fastq-2.0.x-build
export INSTALL=/usr/local/bcl2fastq-2.0.x
```

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-2.0.x.tar.bz2
```

This command creates a bcl2fastq-2.0 sub-directory in the ${TMP} directory.

2 Configuring the build:

```
mkdir ${BUILD}
cd ${BUILD}
${SOURCE}/src/configure --prefix=${INSTALL}
```

3 Building:

```
make
```

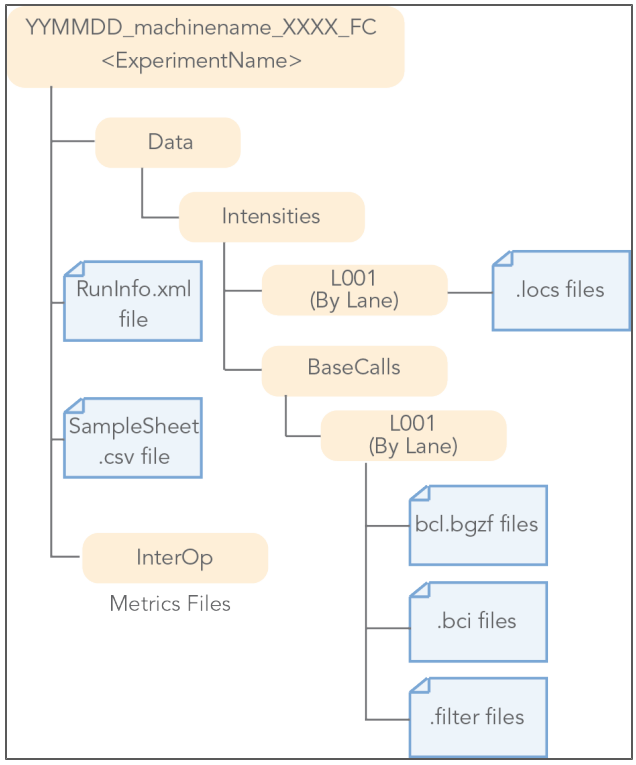
4 Installing (can require root privilege, depending on the ${INSTALL} path):

```
make install
```

Bcl Conversion Input Files

Demultiplexing needs a BaseCalls directory to start a run, while a sample sheet is optional. These files are described in this section.

Figure 2 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
```

Do not deviate from the run folder naming convention. Deviating from the convention can 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 can consist of any combination of upper or lower case letters, digits, or hyphens, but ***cannot*** 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 supplies 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. 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 130108_instrument1_0147 indicates experiment number 147, run on instrument 1, on the 8th of January 2013. 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.

More information is captured in the run folder name in fields separated by an underscore from the first three fields. For example, you can capture the flow cell number in the run folder name as follows: YYMMDD_machinename_NNNN_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 NextSeq.

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

- BCL files (bcl.bgzf)
- *.bci files
- *.filter files
- *.locs files
- RunInfo.xml file. The RunInfo.xml is at the top level of the run folder.
- (Optionally) SampleSheet.csv. The sample sheet is at the top level of the run folder and must be named SampleSheet.csv.

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

BCL Files

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

```
Data/Intensities/BaseCalls/L<lane>
```

They are named as follows:

```
<tile>.bcl.bgzf
```

The BCL files are compressed using the blocked GNU zip format (bgzf). The BCL files are binary base call files with the following format:

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

Bci Files

Bci files contain tile information for the sequencing run in binary format. The Bci files can be found in the BaseCalls directory inside the run directory:

```
Data/Intensities/BaseCalls/L<lane>
```

Bci files are named as follows:

```
s_<Lane>.bci
```

Bci files contain one record per tile, which uses the following format:

- bytes 0-3: tile number
- bytes 4-7: number of clusters in the tile.

Cluster numbers of one bci file sum to a cluster number listed in the beginning of each bcl.bgzf file of that lane.

Filter Files

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

```
Data/Intensities/BaseCalls/L<lane>
```

They are named as follows:

```
s_<Lane>.filter
```

The *.filter files are binary files containing filter results in the following format:

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) Where N is the cluster number	unsigned 8-bits integer: • Bit 0 is pass or failed filter

Position Files

The BCL to FASTQ converter uses *.locs files. The locs files can be found in the Intensities/L<lane> directories.

RunInfo.xml File

The top-level run folder contains a RunInfo.xml file. The file RunInfo.xml (normally generated by the instrument control software) identifies the boundaries of the reads (including index reads).

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

Sample Sheet

The sample sheet (SampleSheet.csv) file details the relationship between samples and indexes specified during library creation. The sample sheet should be located at the top level of the run folder. A sample sheet is not essential; if no sample sheet is provided, all reads are assigned to the default sample Undetermined_S0, one file per lane per read number.

bcl2fastq uses the following sample sheet columns in the [Data] section:

Column	Description
SampleID	ID of the sample
SampleName	Descriptive name of the sample
index	Index sequence
index2	Index sequence for index 2, if using dual indexing

bcl2fastq uses adapter settings in the [Settings] section. If no adapter sequences are provided, no adapter trimming or masking will be done.

Column	Description
Adapter	The adapter sequence that needs to be trimmed. If an AdapterRead2 is provided, this sequence is only used to trim read 1.
AdapterRead2	The adapter sequence that needs to be trimmed in read 2. If not provided, the sequence specified in Adapter is used.
MaskAdapter	The adapter sequence that needs to be masked. If an MaskAdapterRead2 is provided, this sequence is only used to mask read 1.
MaskAdapterRead2	The adapter sequence that needs to be masked in read 2. If not provided, the sequence specified in MaskAdapter is used.

Reads are saved to the FASTQ files named after the samples specified in the sample sheet (see *FASTQ Files* on page 16).

An example of a valid sample sheet opened in Notepad is shown below.

```
[Header],,,,
Investigator Name,Isabelle,,,
Project Name,Kidney23,,,
Experiment Name,Sample 23,,,
Date,11/27/2013,,,
Workflow,GenerateFASTQ,,,
,,,,
[Settings],,,,
MaskAdapter,CGCGTATACGCGTATA,,,
TrimAdapter,GCGCATATGCGCATAT,,,
,,,,
[Data],,,,
SampleID,SampleName,index,index2
AA,AA,AAAAAAAA,AAAAAAAA
CC,CC,CCCCCCCC,CCCCCCCC
GG,GG,GGGGGGGG,GGGGGGGG
TT,TT,TTTTTTTT,TTTTTTTT
XX,XX,ACGTACGT,ACGTACGT
```

Sample Sheet Demultiplexing Scenarios

Bcl conversion and demultiplexing support four scenarios, as directed by the sample sheet:

- No sample sheet found: all reads are placed in the default Undetermined_S0 FASTQ files.
- Sample sheet found, no [Data] section: all reads are placed in the default Undetermined_S0 FASTQ files.

- Sample sheet found, one sample defined without indices: all reads are placed in the sample FASTQ file that is defined in the sample sheet.
- Sample sheet found, samples defined with indices:
  - Reads without a matching index are placed in the default Undetermined_S0 FASTQ files.
  - Reads with a valid index are placed in the sample FASTQ file as defined in the sample sheet.

For each sample, there will be one file per lane per read number (if reads exist for that sample, lane, and read number)

Generating the Sample Sheet

The user-generated sample sheet (SampleSheet.csv file) describes the samples and the indices used. The sample sheet must be located at the top level of the run folder.

Illumina Experiment Manager

Use the Illumina Experiment Manager (IEM) to generate the sample sheet file. You can also use a text editor or Excel but IEM is recommended.

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 www.illumina.com. A MyIllumina account is required.

On a NextSeq instrument, the same library (sample) or library pool will be distributed across all lanes: lanes are not independent with respect to library. bcl2fastq combines reads with matching indices in one sample, and for every sample writes separate files per lane and read number.

A sample sheet can be generated with IEM, using the following choices:

- Instrument: **MiSeq**
- Workflow: **Other** | **Fastq Only**

If you want to set up choices that are not provided by IEM (such as values for MaskAdapter or MaskAdapterRead2), you can open the sample sheet in Excel, and edit it there.

NOTE
bcl2fastq 2.0 will ignore any non-required columns that are included by IEM.

Sample Sheet Columns

bcl2fastq uses the following sample sheet columns in the [Data] section:

Column	Description
`SampleID`	ID of the sample
`SampleName`	Descriptive name of the sample
`index`	Index sequence
`index2`	Index sequence for index 2, if using dual indexing

bcl2fastq uses adapter settings in the [Settings] section. If no adapter sequences are provided, no adapter trimming or masking will be done.

Column	Description
Adapter	The adapter sequence that needs to be trimmed. If an AdapterRead2 is provided, this sequence is only used to trim read 1.
AdapterRead2	The adapter sequence that needs to be trimmed in read 2. If not provided, the sequence specified in Adapter is used.
MaskAdapter	The adapter sequence that needs to be masked. If an MaskAdapterRead2 is provided, this sequence is only used to mask read 1.
MaskAdapterRead2	The adapter sequence that needs to be masked in read 2. If not provided, the sequence specified in MaskAdapter is used.

NOTE
The options MaskAdapter and MaskAdapterRead2 need to be added manually to the sample sheet.

Illegal Characters

SampleID and SampleName field entries 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. The index sequences for the first and second index reads are specified in the Index and Index2 columns.

For example, if a particular sample was associated with the sequence ACCAGTAA in the first index read, and the sequence GGACATGA in the second, the it would have ACCAGTAA in the Index column and GGACATGA in the Index2 column.

Samples Without Index

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

Running Bcl Conversion and Demultiplexing

bcl2fastq is a single standalone binary that should be invoked the following way:

```
nohup /usr/local/bin/bcl2fastq [options]
```

The option that is specified most is `--runfolder-dir`, while `--output-dir` is optional. An example of a command with those options would be:

```
nohup /usr/local/bin/bcl2fastq --runfolder-dir <RunFolder>
   --output-dir <BaseCalls>
```

This produces a set of FASTQ files in the BaseCalls directory. Reads with an unresolved or erroneous index are placed in the Undetermined_S0_ files.

Options for Bcl Conversion and Demultiplexing

The available options for bcl2fastq are listed below.

Main Options

The options used in basic bcl2fastq use cases are `--runfolder-dir` and `--output-dir`; these are explained below.

Option	Description
`--runfolder-dir`	Path to runfolder directory Default: ./
`-o, --output-dir`	Path to demultiplexed output Default: ./Data/Intensities/BaseCalls/

Advanced Options

The options for more advanced bcl2fastq use cases are listed below.

- Directory options:

Option	Description
`-i, --input-dir`	Path to input directory Default: <runfolder-dir>/Data/Intensities/BaseCalls/
`--intensities-dir`	Path to intensities directory If intensities directory is specified, then the input directory must also be specified Default: <input-dir>/../
`--interop-dir`	Path to demultiplexing statistics directory Default: ./InterOp/

- Processing options:

Option	Description
`-r, --loading-threads`	Number of threads used for loading BCL data. Default: 1

Option	Description
`-d, --demultiplexing-threads`	Number of threads used for demultiplexing, Default: 1
`-p, --processing-threads`	Number of threads used for processing demultiplexed data. Default: 1
`-w, --writing-threads`	Number of threads used for writing FASTQ data. This must not be higher than number of samples. Default: 1

If you want to assign multiple threads, use the following guidelines:

- Amount of threads depends on machine: there should be one thread per CPU core plus little more to supply CPU with work, when some of threads are blocked, waiting for something.
- Amount of threads depends on data: it makes no sense to have several threads writing into single file of local hard drive, i.e. at most one writing thread per sample.
- The most CPU demanding stage is the processing step (-p option) and it should be assigned majority of threads.
- The second most CPU demanding stage is the demultiplexing step (-d option).
- Rreading and writing stages are very lightweight and they do not need too many threads working on them, specially when working with local hard drive. Too many threads means too many parallel read/writes, which means too many seeks and that translates to suboptimal performance.

▸ Behavioral options:

Option	Description
`--ignore-missing-bcls`	Assume 'N'/'#' for missing calls
`--create-fastq-for-index-reads`	Create FASTQ files also for index reads
`--barcode-mismatches`	Number of allowed mismatches per index Multiple entries, comma delimited entries, allowed. Each entry is applied to the corresponding index; last entry applies to all remaining indices. Default: 1
`--minimum-trimmed-read-length`	Minimum read length after adapter trimming. bcl2fastq trims down to the value of this parameter below which further adapter match is masked. Default: 32
`--mask-short-adapter-reads`	Maximum number of useful bases in read after adapter trimming for which whole read is masked. Genuinely short reads such as those from smallRNA can be masked but you can limit the length of proper sequence (ACGT) returned, because you know you are looking at 22-base small RNA. The length of the sequence region (non-N) remaining in such a read does not exceed the value of this parameter. Default: 32

▸ General options:

Option	Description
`-h, --help`	Produce help message and exit
`-v, --version`	Print program version information
`-l, --min-log-level`	Minimum log level Recognized values: NONE, FATAL, ERROR, WARNING, INFO, DEBUG, TRACE

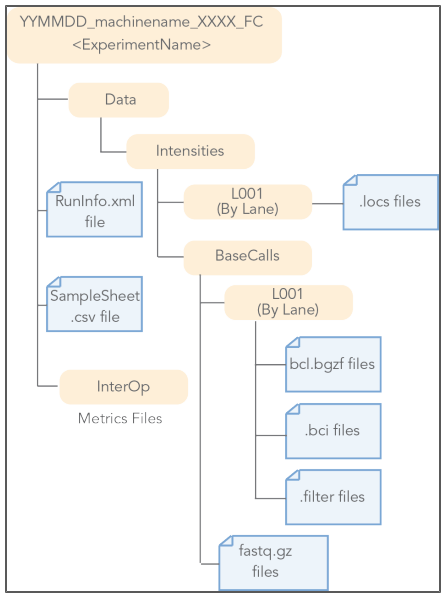
Bcl Conversion Output Folder

The Bcl Conversion output directory has the following characteristics:

- The FASTQ output files are located in the Data/Intensities/BaseCalls directory; names of the files start with the sample name as derived from the sample sheet.
- The Undetermined files contain the reads with an unresolved or erroneous index.
- If no sample sheet exists, the software generates an Undetermined_S0 FASTQ file for each lane and read number.

For each sample, there will be one FASTQ file per lane per read number (if reads exist for that sample, lane, and read number).

Figure 3 Bcl Conversion Output Folder



FASTQ Files

bcl2fastq converts *.bcl files into FASTQ files, which can be used as sequence input for alignment. The files are located in the Data/Intensities/BaseCalls directory. If no sample sheet is provided, the software generates one Undetermined_S0 FASTQ files for each lane and read number combination.

Naming

FASTQ files are named with the sample name and the sample number, which is a numeric assignment based on the order that the sample is listed in the sample sheet. For example:

Data\Intensities\BaseCalls\samplename_S1_L001_R1_001.fastq.gz

- **samplename**—The sample name provided in the sample sheet. If a sample name is not provided, the file name includes the sample ID, which is a required field in the sample sheet and must be unique.
- **S1**—The sample number based on the order that samples are listed in the sample sheet starting with 1. In this example, S1 indicates that this sample is the first sample listed in the sample sheet.

NOTE
Reads that cannot be assigned to any sample are written to a FASTQ file for sample number 0, and excluded from downstream analysis.

- **L001**—The lane number.
- **R1**—The read. In this example, R1 means Read 1. For a paired-end run, there is at least one file with R2 in the file name for Read 2.
- **001**—The last segment is always 001.

Compression

FASTQ files are saved compressed in the GNU zip format (an open source file compression program), 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 only 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>:<sample
   number>
```

The following table describes the elements:

Element	Requirements	Description
@	@	Each sequence identifier line starts with @
<instrument>	Characters allowed: a–z, A–Z, 0–9 and underscore	Instrument ID
<run number>	Numerical	Run number on instrument
<flowcell ID>	Characters allowed: a–z, A–Z, 0–9	
<lane>	Numerical	Lane number
<tile>	Numerical	Tile number
<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.
<sample number>	Numerical	Sample number from samplesheet

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

```
@SIM:1:FCX:1:15:6329:1045 1:N:0:2
TCGCACTCAACGCCCTGCATATGACAAGACAGAATC
+
<>;##=><9=AAAAAAAAA9#:<#<;<<<????#=
```

Control Values

If the read is not identified as a control, then the tenth column (`<control number>`) is zero. If the read is identified as a control, the number is greater than zero, and the value specifies what type of control it is. The value is the decimal representation of a bit-wise encoding scheme. In that scheme bit 0 has 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 quality score, Q(A), expresses the probability that A is not true, P(~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

In FASTQ files, quality scores are encoded into a compact form, which uses only 1 byte per quality value. In this encoding, the quality score is represented as the character with an ASCII code equal to its value + 33. The following table demonstrates the relationship between the encoding character, its ASCII code, and the quality score represented.

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

Symbol	ASCII Code	Q-Score	Symbol	ASCII Code	Q-Score	Symbol	ASCII Code	Q-Score
&	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			

InterOp Files

The Interop files can be found in the directory: <run directory>\InterOp, and contains binary files used by Sequencing Analysis Viewer (SAV) to summarize various primary analysis metrics such as cluster density, intensities, quality scores, and overall run quality.

The index metrics are stored in the file *IndexMetricsOut.bin*, which has the following binary format:

- Byte 0: file version (1)
- Bytes (variable length): record:
  - 2 bytes: lane number (unint16)
  - 2 bytes: tile number (unint16)
  - 2 bytes: read number (unint16)
  - 2 bytes: number of bytes Y for index name (unint16)
  - Y bytes: index name string (string in UTF8Encoding)
  - 4 bytes: # clusters identified as index (uint32)
  - 2 bytes: number of bytes V for sample name(unint16)
  - V bytes: sample name string (string in UTF8Encoding)
  - 2 bytes: number of bytes W for sample project(unint16)
  - W bytes: sample project string (string in UTF8Encoding)

Appendix: Requirements

Network Infrastructure

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

Server Configurations

You can use a single multi-processor or a multi-core computer running Linux.

Analysis Computer

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

Memory Requirements

bcl2fastq requires a minimum of 16 GB RAM per core.

Software Requirements

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

The following software is required to run bcl2fastq:

- zlib
- librt
- libpthread

To build bcl2fastq, you need the following software. Versions listed are tested and supported; newer versions are untested.

- gcc 4.1.2 (with c++)
- boost 1.54 (with its dependencies)
- cmake 2.8.9
- zlib
- librt
- libpthread

Technical Assistance

For technical assistance, contact Illumina Technical Support.

Table 2 Illumina General Contact Information

Illumina Website	www.illumina.com
Email	techsupport@illumina.com

Table 3 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

Safety Data Sheets

Safety data sheets (SDSs) are available on the Illumina website at www.illumina.com/msds.

Product Documentation

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