SOLiD™ System XSQ Tools

Contents

For Research Use Only. Not intended for any animal or human therapeutic or diagnostic use.

January 2012

Overview

As sequencing technologies evolve and their output capacity increases, large amounts of new sequence data are straining current data storage and analysis systems in the genomics community. One approach to mitigating this pain is to store the instrument data in a more compact format, replacing the older text based formats — FASTQ, CSFASTA, and QUAL — with more efficient binary encoding. The Extensible Sequence (XSQ) file format has been developed to store each call and quality value in a single byte, which results in file sizes that are up to 75% smaller. There are many other benefits of this format, including embedded metadata, hierarchical structure of the file allowing simplified access to its contents, and the ability to store parallel datasets for easier integration of results.

Each new data format, however, comes with costs. Learning the intricacies of a new format can be easier with full descriptions and simplified examples. Integrating the new format with legacy data or legacy workflows can be more challenging, but this can be mitigated through the use of appropriate data format converters. The goal of the XSQ Tools package is to lower the activation energy associated with adopting the XSQ format by providing documentation, examples, and converters.

Utilities in the XSQ Tools package perform these functions:

- For SOLiD™ 4 System users, convert CSFASTA format files into the XSQ format required by LifeScope™ Software
- For 5500 Series SOLiD™ Sequencer users running BioScope™ Software or third-party tools, convert XSQ files into the CSFASTA format required by BioScope™ Software
- For LifeScope™ Software users, split an XSQ file into multiple XSQ files, each containing the data for one library
- Change the index (barcode number) of an XSQ file

More information about XSQ files and the XSQ file format is available on the SOLiD™ Software Tools web site:

http://solidsoftwaretools.com/gf/project/xsq

This site provides:

- The XSQ file format specification document
- Slides describing XSQ and its relation to the 5500 Series SOLiD™ Sequencer
- A link to the XSQ Tools package

The following table lists the platforms supported by the XSQ Tools package.

Supported platforms for the XSQ conversion tools

Platform	Conversion type	
	To XSQ	From XSQ
Linux® CentOS 4.7 or later	Yes	No
Linux® CentOS 5.5 or later	Yes	Yes

The XSQ Tools package

Package components

The following components are included in the XSQ Tools package:

- The `convertToXSQ.sh` wrapper script, which sets paths and executes tools for converting from CSFASTA and QUAL to XSQ
- The `convertFromXSQ.sh` wrapper script, which sets paths and executes tools for the following:
  - Conversion from XSQ to CSFASTA and QUAL
  - Conversion from XSQ to FASTQ
  - Splitting an indexed (barcoded) XSQ file by library into new XSQ files
- The `indexReassignment.sh` wrapper script, which supports library index reassignment for an XSQ file
- Operating system-specific software libraries to support standalone executables
- Example data sets, which show the data format using simulated reads. We recommend you use the example data to verify your setup and to practice a conversion. This data is based on chromosome 6.
  Note: These do not represent data generated by an instrument.
- Example usage scripts
- End User License Agreement (EULA)
- Third-party licenses
- User guide

Download instructions

To download the XSQ Tools package, visit the following site:

http://solidsoftwaretools.com/gf/project/xsq

You must accept the EULA and follow the download directions on the site. After downloading the file, follow these steps to install the XSQ Tools package:

1. Move the file to an appropriate location.
2. Change directory to the tarball location.
3. Execute the command `tar xvfz XSQ_Tools.tgz`. The package is untarred in a directory named `XSQ_Tools`.
4. Add the `XSQ_Tools` directory to your path, with one of the following commands:

   bash: `export PATH=$PATH:$`*tarlocation*`/XSQ_Tools`

   csh/tcsh: `set PATH = ( $PATH:$`*tarlocation*`/XSQ_Tools )`

   where *tarlocation* is the directory containing the package tarball.

Resolved issues

The following issues are resolved in the August 2011 release:

- Splitter failure due to incorrect LibraryDetails format has been fixed. The fields `IndexID` and `IndexName` must each be unique and have a 1:1 mapping.
- Converter failure due to missing libraries has been fixed. The converter now handles the scenario of a library group present in the LibraryDetails table but missing from the XSQ file.
- Platform checking now accepts later versions of supported systems.

Conversion to the XSQ format

This section describes converting CSFASTA and QUAL files from earlier SOLiD™ Systems to the XSQ format. This conversion is also supported in the LifeScope™ Software UI.

The `convertToXSQ.sh` script converts a pair of CSFASTA and QUAL files to an XSQ file. The command options vary by library type, as shown below:

- To convert SOLiD™ fragment data:

```
sh convertToXSQ.sh \
    --mode=Fragment \
    --c1=<csfastafile> --q1=<qualfile> \
    --xsqfile=<output XSQ path and filename> \
    --libraryName=<LibraryName> \
    --laneNumber=<LaneNumber> \
    --runStartTime="yyyy-mm-dd hh:mm:ss"
```

- To convert SOLiD™ mate-pair *or* paired-end data:

```
sh convertToXSQ.sh \
    --mode=LMP  OR  --mode=PE \
    --c1=<csfastafile> --q1=<qualfile> \
    --c2=<csfastafile> --q2=<qualfile> \
    --libraryInsertSizeMinimum=<Minimum insert size> \
    --libraryInsertSizeMaximum=<Maximum insert size> \
    --xsqfile=<output XSQ path and filename> \
    --libraryName=<LibraryName> \
    --laneNumber=<LaneNumber> \
    --runStartTime="yyyy-mm-dd hh:mm:ss"
```

- To convert SOLiD™ barcoded fragment data (a single barcoded fragment data set):

```
sh convertToXSQ.sh \
    --mode=BC \
    --c1=<csfastafile> --q1=<qualfile> \
    --bc1=<index csfastafile> --bq1=<index qualfile> \
    --xsqfile=<output XSQ path and filename> \
    --libraryName=<LibraryName> \
    --laneNumber=<LaneNumber> \
    --runStartTime="yyyy-mm-dd hh:mm:ss"
```

- To convert SOLiD™ barcoded paired-end data (a single barcoded paired-end data set):

```
sh convertToXSQ.sh \
    --mode=BCPE \
    --c1=<csfastafile> --q1=<qualfile> \
    --c2=<csfastafile> --q2=<qualfile> \
    --bc1=<index csfastafile> --bq1=<index qualfile> \
    --libraryInsertSizeMinimum=<Minimum insert size> \
    --libraryInsertSizeMaximum=<Maximum insert size> \
```

```
--xsqfile=<output XSQ path and filename> \
--libraryName=<LibraryName> \
--laneNumber=<LaneNumber> \
--runStartTime="yyyy-mm-dd hh:mm:ss"
```

The options required by the `convertToXSQ.sh` script are listed in the following table.

Required arguments for the `convertToXSQ.sh` script (to create an XSQ file)

Option	Required with...	Description
--mode	All	Allowed values are Fragment, LMP, PE, BC, BCPE: • **Fragment**: Convert fragment CSFASTA and QUAL files to XSQ format. • **LMP**: Convert mate-pair CSFASTA and QUAL files to XSQ format. • **PE**: Convert paired-end CSFASTA and QUAL files to XSQ format. • **BC**: Convert single barcoded fragment CSFASTA/QUAL with index information to XSQ format. • **BCPE**: Convert single barcoded paired-end CSFASTA/QUAL with index information to XSQ format.
--libraryName	All	Name of the library preparation.
--laneNumber <*arg*>	All	Lane number in which this sequencing sample was loaded within the flowchip layout.
--runStartTime <*arg*>	All	Time when the instrument run was executed. This information is entered into the output XSQ file. Use one of these formats: • `'yyyy-mm-dd hh:mm:ss'` (including the quotes, either double or single). • `yyyy-mm-dd\ hh:mm:ss` (escape the space with a backslash).
--xsqfile <*arg*> *or* -x <*arg*>	All	Requested name of output XSQ file. The name must include the extension `.xsq`.
--c1 <*arg*>	All SOLiD™ System data	The CSFASTA input file for the first tag.
--q1 <*arg*>	All SOLiD™ System data	The QUAL input file for the first tag.
--c2 <*arg*>	LMP, PE, or BCPE SOLiD™ System data	The CSFASTA input file for the second tag.
--q2 <*arg*>	LMP, PE, or BCPE SOLiD™ System data	The QUAL input file for the second tag.

--bc1 <*arg*>	BC or BCPE SOLiD™ System data	The index (barcoded) CSFASTA input file.
--bq1 <*arg*>	BC or BCPE SOLiD™ System data	The index (barcoded) QUAL input file.
libraryInsertSizeMinimum	LMP or PE SOLiD™ System data	The expected minimum insert size, in nucleotides.
libraryInsertSizeMaximum	LMP or PE SOLiD™ System data	The expected maximum insert size, in nucleotides.

The following table lists options for the `convertToXSQ.sh` script.

Optional arguments for the `convertToXSQ.sh` script (to create an XSQ file)

Option		Description
-a <*arg*>	--application <*arg*>	The type of analysis intended for this input data. Allowed values are listed below. Quotes are required around values that contain spaces. • Whole genome resequencing • Targeted resequencing • Whole transcriptome (Fragment) • Whole transcriptome (PairedEnd) • Small RNA • ChIP-Seq • Methylation
	--assembly <*arg*>	The intended assembly for mapping. May be blank if unknown. Examples: • hg18 • hg19 • mm9 Other assembly names are allowed. Spaces are not allowed in these names.

-c <*arg*>		Use the configuration file <*arg*> as a list of conversion parameters. The parameters are specified as nested key value pairs. Example file content:
		`q1=/data/results/reads/uhr.150.F3.qual`
		`c1=/data/results/reads/uhr.150.F3.csfasta`
		`x=./out.xsq`
		`mode=Fragment`
		`runStartTime=2011-04-01\ 12:01:02`
		`libraryName=150_UHR`
		Option hyphens ('-', '--') are not used in the configuration file.

	--comments	Additional information about this sample or its preparation.
-d <*arg*>	--description <*arg*>	A description of the sample.
-e	--ercc	Whether internal controls have been used. Reserved for future use.
	--flowCellAssignment <*arg*>	The position of the instrument flowcell in which this flowchip was placed.
-h	--help	Display command usage and options.
	--instrumentName <*arg*>	The name of the instrument.
	--instrumentSerial <*arg*>	The unique serial number identifying the instrument.
-k <*arg*>	--indexKitName <*arg*>	The name of the kit used to add the indexing nucleotides.
	--libraryName <*arg*>	The name of library preparation.
-n	--noqualfile	Use this attribute to specify that the QUAL file is not available.
-o <*arg*>	--operator <*arg*>	The name or ID of person who set up the instrument run.
-p <*arg*>	--projectName <*arg*>	Project within which this sample is being run. This value is entered into the projectName field in the libraryDetails table, and may be different from the LifeScope™ Software project name.
	--runEndTime <*arg*>	Time when the instrument run completed. This information is entered into the output XSQ file. Use one of these formats: • `'yyyy-mm-dd hh:mm:ss'` (including the quotes, either double or single). • `yyyy-mm-dd\ hh:mm:ss` (escape the space with a backslash).
	--runName <*arg*>	The name of a flowchip run.
	--sampleIdentifier <*arg*>	The unique name for this sample.
	--sampleOwner <*arg*>	The owner of this sample.
	-sequencingSampleDescription <*arg*>	Description of the sample loaded in a lane on a flowchip.
	--sequencingSampleName <*arg*>	The name of the sample loaded in a lane on a flowchip.
	--species <*arg*>	The intended species for mapping. Example values include: • Homo sapiens • Mus musculus • other

-V	--version	Print version information.

Conversion from the XSQ format

This section describes converting XSQ format files from the 5500 Series SOLiD™ Sequencer into CSFASTA and QUAL files, in the format required by earlier version of BioScope™ Software. This section does not apply to LifeScope™ Software users who are analyzing XSQ data files generated by the 5500 Series SOLiD™ Sequencer.

The conversion program is available as a standalone script to be run on the Linux shell. This conversion is not available within the LifeScope™ Software UI or the LifeScope™ Software command shell.

If the input XSQ file includes base-space data, the conversion also exports the base-space data into a FASTQ file.

Conversion syntax

The `convertFromXSQ.sh` script converts an XSQ file into a pair of CSFASTA and QUAL files. The usage for this script is:

```
convertFromXSQ.sh -c xsqfile
```

where *xsqfile* gives an existing XSQ file on the file system.

The converted files are created in the following directory:

./Libraries/<*LibraryName*>/<*TagName*>/reads/

The output files are named according to the following pattern:

<xsqfile>_<LibraryName>_<TagName>.ext

where *ext* has the following values:

- `.csfasta`
- `.QV.qual`
- `.fastq`

FASTQ files are created only if the input XSQ file contains base-space data.

Filter option

Use the -f option to filter (remove) marked reads during conversion, with this version of the command:

```
convertFromXSQ.sh -c –f xsqfile
```

In the XSQ file, each read contains a field of filtering flags, which may mark the read for filtering. Only when the -f option is used, reads that are marked for filtering are not written to the CSFASTA file.

By default convertFromXSQ.sh ignores the filtering flags and writes all reads to the converted output files. With the -f option, the converter applies the filtering flags during conversion, which suppresses the filtered reads and quality values from being written in the output files.

Options table

The following table lists the options supported by the `convertFromXSQ.sh` script. These options include functionality for XSQ file-splitting and for version information.

Options for the `convertFromXSQ.sh` script

Option	Description
-c	Converts the input XSQ file into CSFASTA, QUAL, and FASTQ files.
-f	(*Optional*) During conversion, removes reads that are marked for filtering.
-o	(*Optional*) Specifies an output directory partial path.
-s	Splits the input XSQ file into multiple output files, one for each library in the parent XSQ file.
-v	Displays the package version.

XSQ file splitting

The `convertFromXSQ.sh` script with the `-s` option splits an input XSQ file into multiple XSQ files, one file per library. For example, if a paired-end library is created with indices for each of `lib1`, `lib2`, and `lib3`, and run in the same lane, a single XSQ file will be created. Using the `-s` option creates three separate XSQ files - one for each of `lib1`, `lib2`, and `lib3`. Each XSQ file contains data for both the F3 and F5 tags:

```
convertFromXSQ.sh -s xsqfile -o outputpath
```

The following output files are produced:

```
outputpath/
    lib1.xsq
    lib2.xsq
    ...
```

Only one input XSQ file is allowed per run.

XSQ indexing reassignment

Purpose

The `indexReassignment.sh` script supports correcting the index information in an XSQ file (for example, in case incorrect information is entered on the instrument). The process involves these steps:

1. Run the `indexReassignment.sh` script to write a comma-separated values (CSV) file.
2. Manually edit the CSV file to enter the correct index name.

3. Run the `indexReassignment.sh` script to read the CSV file and create an XSQ file with the corrected index information.

The `indexReassignment.sh` script also supports optionally changing the mismatch level.

Usage

This section describes the usage of the `indexReassignment.sh` script.

- To *extract* the index information:

  ```
  indexReassignment.sh –e –xsq example.xsq
  ```

  This command creates a CSV file with the same file name as the input XSQ file but with a `.csv` extension. (You then manually edit the CSV file to correct the index information, before running `indexReassignment.sh` to update the information.)

- To *update* with the corrected index information:

  ```
  indexReassignment.sh –csv example.csv –xsq example.xsq –o example-fixed.xsq
  ```

  This command creates a new XSQ file with the corrected index information. The options have the following meaning:

  `-csv <arg>`: The CSV file edited to correct the index.

  You create the CSV file during the `indexReassignment.sh` extraction step above, and manually edit it to correct the index.

  `-xsq <arg>`: The original XSQ file.

  `-o <arg>`: The output XSQ file, newly-created with the corrected index information.

- To *change the mismatch* level while updating the index:

  ```
  indexReassignment.sh –csv example.csv –xsq example.xsq –o example-fixed.xsq –mm n
  ```

  The `–mm n` option changes the update command to also update the mismatch level. This option redoes the entire indexing assignment and rewrites all library groups. (Allowed values with the `–mm` option are 0 and 1.)

Index reassignment options table

The following table lists the options supported by the `indexReassignment.sh` script.

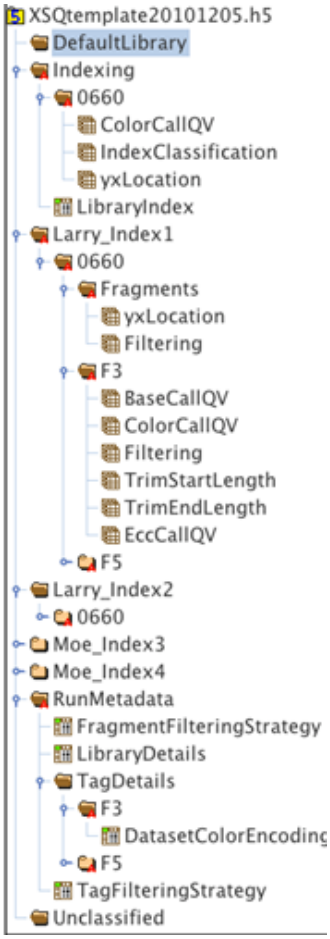
Options for the `indexReassignment.sh` script

Option	Description
-csv <*arg*>	Update the XSQ file index, reading index information from the CSV file <*arg*>
-e	Extract current index information
-mm <*arg*>	Mismatch level for index classification
-o <*arg*>	Output XSQ file, created with the corrected index information

-xsq <*arg*>	Input XSQ file

HDF5 tools

The XSQ format conforms to the HDF5 specification, and HDF5 tools work on XSQ files. The following figure is an example of how HDFView displays an XSQ file, showing Groups, Tables, and Datasets.



HDFView screenshot of the XSQ file format, showing Groups, Tables, and Datasets

XSQ file format

This section describes several tables within an XSQ file. The complete XSQ file format specification is available on the SOLiD™ Software Tools web site. At the time of this writing the specification is this site:

http://solidsoftwaretools.com/gf/project/xsq

The LibraryDetails table describes the properties of each library used in the XSQ file. String fields are limited in length to 255 characters.

XSQ library details table

	Column name	Description
1	LibraryName	Describes the name of one library preparation, which may be used across several indexes in an indexing run.
2	Application	Describes how this sequence is intended to be analyzed. Allowed values are: • Whole genome resequencing • Targeted resequencing • Whole transcriptome (Fragment) • Whole transcriptome (PairedEnd) • Small RNA • ChIP-Seq • Methylation
	ProjectName	Describes the project within which this sample is being run.
3	SampleOwner	Names the owner of this sample.
4	SampleIdentifier	A unique name for this sample.
5	Description	Describes the sample.
6	Comments	Optional additional information about this sample or its preparation.
7	ERCC	Whether internal controls have been used. Reserved for future use.
8	Species	The intended species for mapping. Example values include: • Homo sapiens • Mus musculus • other
9	Assembly	The intended assembly for mapping (may be blank if unknown). Example values include: • hg18 • hg19 • mm9 Other valid assembly names are accepted. Spaces are not allowed.
10	LibraryInsertSizeMinimum	The expected minimum insert size in nucleotides. Used with paired-end and mate-pair libraries. Allowed values: Integers >= 0. Set to 0 for fragment libraries, or when the size is unknown.

Each read type (tag) has tag-level properties in the XSQ file. The following table lists the attributes describing the base-space and color-space data for a single type of read.

XSQ tag details table

	Column name	Default	Description
1	TagSequence	—	The full sequence of the tag used for primer annealing. It is used in conjunction with the read offset to determine the nucleotide or nucleotides for phasing the color-space reads. When the full sequence is not available, the last base(s) of the tag are sufficient as long as they cover the largest offset in the DatasetColorEncoding table. Conditional: Required when color space is present.
2	IsColorPresent	0	Whether color-space call data is present. Values: • 0: No calls are reported in color space (causing the ColorCallQV dataset to be missing). • Non-zero: Color calls are reported.
3	IsBasePresent	1	Whether base call data is present. Values: • 0: No calls are reported in base space (causing the BaseCallQV dataset to be missing). • Non-zero: Base calls are reported.
4	NumBaseCalls	—	The maximum untrimmed read length in base space across all reads for this tag. This length is the width of the BaseCallQV dataset. This field is not used when the BaseCallQV dataset is not present, but is required when BaseCallQV is present.
5	MinTrimmedReadLength	—	The minimum trimmed read length across all unfiltered reads for this tag. Allowed values: Integers >= 0.
6	DoesRepresentedStrandMatchSource	1	Whether the reads are represented on the same strand as the source. Values: • 0: The synthesis strand is not the same as the source strand. • Non-zero: The synthesis strand is the same as the source strand.
7	RelativeOrder	—	Describes the relative order of a read on the sense strand of the source nucleic acid, in relation to all of the other reads from the same fragment. 1 based. Required only when there is more than one tag per fragment.
8	SynthesisFromThreePrime	0	(*Optional*) Whether the synthesis direction of the synthesized strand is from the 3′ end. Values: • 0 or missing: synthesis from the 5′ end to the 3′ end. • Non-zero: The synthesis direction is from the 3′ end to the 5′ end.

The following table lists tag names for current library types.

Tag information for current library types

Instrument	Library type	Indexing	Tag name	Base	Does Represented-StrandMatch-Source	Relative-Order	Synthesis-FromThree-Prime
SOLiD™	Fragment	Yes | no	F3	T	1	1	1
SOLiD™ and 5500	Mate-pair	No	F3	T	1	2	1
			R3	G	1	1	1
SOLiD™	Paired-end	No	F3	T	1	1	1
			F5-P2	T	0	2	0
SOLiD™	Paired-end	Yes	F3	T	1	1	1
			F5-BC	G	0	2	0
5500	Paired-end (DNA)	Yes | no	F3	T	1	1	1
			F5-DNA	T	0	2	0
5500	Paired-end (RNA)	Yes | no	F3	T	1	1	1
			F5-RNA	G	0	2	0
Illumina®	Paired-end	—	Forward	—	0	1	0
			Reverse	—	1	2	0

The DatasetColorEncoding table in the XSQ file is required when color space is present. The table describes how reads are encoded as colors by the instrument. Indexing reads are not included in this table. The XSQ DatasetColorEncoding table is described in the following table.

The DatasetColorEncoding table

	Column name	Description
1	DataSetName	Must match the name of a read dataset under a tag group. This table

		describes that read dataset.
2	Offset	Describes the distance and direction of the primer end as compared to the tag end. The offset is used to relate individual calls to the bases they encode.
3	Encoding	Contains a set of integers that direct the conversion from nucleotides to colors as described below. (2BE is `11`, 4BE is `1303`).
4	Stride	Used to pack discontinuous reads into a continuous array. Stride is used to determine the relative positioning of the start nucleotides between color calls. The use of stride disallows collating calls from multiple offsets into a single call vector when an incomplete set is used, so these are separated into different datasets.
5	NumColorCalls	The maximum untrimmed read length in base space across all reads for this tag. This value is the width of the associated CallQV dataset.

Internal conversion parameters

The following table lists and describes the parameters used internally by the LifeScope™ Software UI when converting a file to the XSQ format from the older CSFASTA and QUAL formats used in earlier versions of the software. These parameters are for internal use.

Internal XSQ conversion parameters

Parameter	Default	Description
xsqconverter.run	—	For internal use.
xsq.application	—	Describes how this sequence is intended to be analyzed. Allowed values are: • Whole genome resequencing • Targeted resequencing • Whole transcriptome (Fragment) • Whole transcriptome (PairedEnd) • Small RNA • ChIP-Seq • Methylation

xsq.assembly	—	The intended assembly for mapping. May be blank if unknown. Example values: • hg18 • hg19 • mm9
xsq.comments	—	Additional information about this sample or its preparation.
xsq.description	—	Sample description.
xsq.ercc	0	Whether or not internal controls have been used. Allowed values: • **0**: No • **1**: Yes
xsq.flowCellAssignment	255	The position of the instrument flowcell in which this flowchip was placed.
xsq.indexKitName	—	Name of the kit used to add the indexing nucleotides.
xsq.input.barcodedCsfasta1	—	Barcoded CSFASTA file (used only in the case of indexed runs).
xsq.input.barcodedQual1	—	Barcoded QUAL file (used only in the case of indexed runs).
xsq.input.csfasta1	—	Input CSFASTA file.
xsq.input.csfasta2	—	Second CSFASTA file (use only in the case of LMP or PE runs).
xsq.input.qual1	—	Input QUAL file.
xsq.input.qual2	—	Second QUAL file (use only in the case of LMP or PE).
xsq.instrumentName	—	Name of the sequencing instrument.
xsq.instrumentSerial	—	Unique serial number to identify a particular sequencing instrument.
xsq.laneNumber	255	Lane number within the flowchip layout in which this sequencing sample was loaded.
xsq.libraryInsertSizeMaximum	—	Expected maximum insert size in nucleotides. (Required for LMP and PE conversion).
xsq.libraryInsertSizeMinimum	—	Expected minimum insert size in nucleotides. (Required for LMP and PE conversion).
xsq.libraryName	—	The name of library preparation.

xsq.mode	—	XSQ mode of operation. Allowed values are: • **Fragment**: Convert fragment CSFASTA and QUAL formats to XSQ. • **LMP**: Convert mate-pair CSFASTA and QUAL formats to XSQ. • **PE**: Convert paired-end CSFASTA and QUAL formats to XSQ. • **BC**: Converting single barcoded fragment CSFASTA and QUAL formats with index information to XSQ. • **BCPE:** Convert single barcoded paired-end CSFASTA and QUAL formats with index information to XSQ.
xsq.noqualfile	false	Specify that the input QUAL file is not available. Allowed values: • **false**: The QUAL file is available. • **true**: The QUAL file is missing.
xsq.operator	—	The name of person or ID who set up the instrument run.
xsq.output.xsqfile	—	The requested name for output XSQ file. Must use the ".xsq" extension.
xsq.projectName	—	Project within which this sample is being run.
xsq.runEndTime	—	Time when the instrument run completed. Required format: `'yyyy-mm-dd hh:mm:ss'` (including quotes).
xsq.runName	—	Name of a flowchip run.
xsq.runStartTime	—	Time when the instrument run started. Required format: `'yyyy-mm-dd hh:mm:ss'` (including quotes).
xsq.sampleIdentifier	—	Unique name for this sample.
xsq.sampleOwner	—	Owner of this sample.
xsq.sequencingSampleDescription	—	Description of the sample loaded in a lane on a flowchip.
xsq.sequencingSampleName	—	Name of sequencing sample loaded in a lane on a flowchip. This value is the bead emulsion loaded on the instrument. The emulsion may be a collection of barcoded libraries from multiple samples.
xsq.species	—	The Intended species for mapping. Example values include: • Homo sapiens • Mus musculus • Other

Internal INI file

The following is an example of an INI file used internally by the LifeScope™ Software UI during a conversion to the XSQ format from the older CSFASTA and QUAL formats used in earlier versions of the software. This INI file is for internal use.

```
##Run parameter
xsqconverter.run=1

##How this sequences is intended to be analyzed
xsq.application=Unknown

##Intended assembly for mapping
xsq.assembly=Unknown

##Additional information about this sample or its preparation
xsq.comments=Unknown

##Sample description
xsq.description=Unknown

##Whether internal controls have been used (0=No, 1=Yes)
xsq.ercc=0

##The position of the instrument flowcell in which this flowchip was placed
xsq.flowCellAssignment=255

##Name of the kit used to add the indexing nucleotides
xsq.indexKitName=Unknown

##Barcoded csfasta file (should be used only in case of indexed runs)
xsq.input.barcodedCsfasta1=

##Barcoded qual file  (should be used only in case of indexed runs)
xsq.input.barcodedQual1=

##Csfasta file
xsq.input.csfasta1=

##Second csfasta file (should be used only in case of LMP or PE)
xsq.input.csfastac2=

##Qual file
xsq.input.qual1=

##Second qual file (should be used only in case of LMP or PE)
xsq.input.qual2=

##Name of instrument
xsq.instrumentName=Unknown

##Unique serial number to identify a particular instrument
xsq.instrumentSerial=Unknown

##Lane number within the flowchip layout in which this sequencing sample was loaded
xsq.laneNumber=255

##Expected maximum insert size in nucleotides. (Required for LMP and PE conversion)
xsq.libraryInsertSizeMaximum=
```

```
##Expected minimum insert size in nucleotides. (Required for LMP and PE conversion)
xsq.libraryInsertSizeMinimum=

# The name of library preparation
xsq.libraryName=DefaultLibrary

# Allowed values are: Fragment, LMP, PE, BC, and BCPE. Use 'Fragment' for converting
fragment csfasta/qual to XSQ. Use 'LMP' for converting mate pair csfasta/quals to XSQ.
Use 'PE' for converting paired end csfasta/quals to XSQ. Use 'BC' for converting  single
barcoded fragment csfasta/qual with index information to XSQ. Use 'PE' for converting
single barcoded paired end csfasta/qual with index information to XSQ
xsq.mode=Fragment

##Specify that qual file is not available (Use 'true' - to specify qual file missing)
xsq.noqualfile=false

##The name of person/ID who set up the instrument run
xsq.operator=Unknown

##Desired name for output XSQ file
xsq.output.xsqfile=pluginfrag.xsq

##Project within which this sample is being run
xsq.projectName=Unknown

##Time when run completed. The run end time should be in 'yyyy-mm-dd hh:mm:ss' format
xsq.runEndTime=

##Name of a flowchip run
xsq.runName=Unknown

##Time when run was executed. The run start time should be in 'yyyy-mm-dd hh:mm:ss'
format
xsq.runStartTime=2011-01-11 20:10:20

##Unique name for this sample
xsq.sampleIdentifier=Unknown

##Owner of this sample
xsq.sampleOwner=Unknown

##Description of the sample loaded in a lane on a flowchip
xsq.sequencingSampleDescription=Unknown

##Name of sample loaded in a lane on a flowchip
xsq.sequencingSampleName=Unknown

##Intended species for mapping
xsq.species=Unknown
```

Links to resources

This section lists resources related to the XSQ file format and converters.

For the XSQ Tools package and documentation, XSQ file format specification, XSQ webinar slides, and example XSQ files, visit this site:

http://solidsoftwaretools.com/gf/project/xsq

HDFView is a Java® tool for browsing HDF-based files, including XSQ files. For information about HDFView, visit this site:

http://www.hdfgroup.org/hdf-java-html/hdfview/

HDF5 is a technology suite for managing of large, complex data collections. HDF5 APIs are available at this site:

http://www.hdfgroup.org/HDF5/release/obtain5.html

Exact Call Chemistry (ECC) is a unique ligation-based sequencing methodology, introduced with the 5500 Series SOLiD™ Sequencer. The Exact Call Chemistry white paper is available through this link and the instructions that follow:

http://www.appliedbiosystems.com/absite/us/en/home/applications-technologies/solid-next-generation-sequencing/publications-literature.html

1. Under the Publication & Literature section, click the Product Literature tab.
2. Scroll down to the White Papers section.
3. Click on the "SOLiD™ System Accuracy with the Exact Call Chemistry Module" link.

FAQ

1

What software accepts the XSQ file format?

Initially, only LifeScope™ Software supports the new format. Life Technologies Corporation is working with third-party developers to adapt their workflows to support the new chemistry and data format.

2

I have pipelines that require CSFASTA and QUAL files as input. What should I do?

Life Technologies Corporation provides tools on the SOLiD™ Software Tools web site to convert XSQ files into CSFASTA and QUAL files. See Links to resources, and download the XSQ Tools package.

3

I have pipelines that require FASTQ files as input. What should I do?

When the ECC module is used during sequencing, base-space data is available in the XSQ file and can be exported into a FASTQ file. See Conversion from the XSQ format.

4

Can I use data from both a SOLiD™ 4 System and a 5500 Series SOLiD™ Sequencer for data analysis?

Yes. There are two options, and both involve converting the data into a common format, either XSQ or CSFASTA+QUAL. One way is to convert the SOLiD™ 4 System CSFASTA and QUAL output to the XSQ format (see Conversion to the XSQ format). Then supply both the converted XSQ file and the 5500™ XSQ file as input to your LifeScope™ 2.0 analysis.

Another method is to convert the XSQ data into CSFASTA+QUAL for use with older analysis tools.

5

Is there a converter to change CSFAST and QUAL files to the XSQ format?

Yes, standalone converters are provided in the XSQ Tools package at the following site:

http://solidsoftwaretools.com/gf/project/xsq

6

How does the XSQ format handle multiplexing?

Multiple libraries may be included in a single XSQ file, such as are generated for each lane of a SOLiD™ 5500 Series Sequencer run. When multiple different users are providing samples to be run in the same lane, the resulting file may be split by library for distribution to individual users, without sharing all of the data with each user. The `convertFromXSQ.sh` script provides the -s splitting option to separate libraries into separate XSQ files when necessary. See XSQ file splitting.

Note: The -s splitting option only works for indexing files. Non-indexing files already contain only a single library (and are not split with the -s option).

The `convertToXSQ.sh` script is run once for each barcode index, and each run results in a single XSQ file. Multiple barcode indices cannot be joined in a single XSQ file with the current converter.

7

Are APIs available for accessing XSQ data?

Yes. HDF5 APIs work with the XSQ format, and are available at this site:

http://www.hdfgroup.org/HDF5/release/obtain5.html

8

Is an XSQ file viewer available?

Yes. XSQ data is can be viewed with the HDFView browser, which can be obtained at this site:

http://www.hdfgroup.org/hdf-java-html/hdfview/

9

Can XSQ files be converted in the LifeScope™ Software UI and command shell?

Yes. The LifeScope™ Software UI supports the conversion of CSFASTA and QUAL files to the XSQ format. If you are running the LifeScope™ Software command shell, use the standalone XSQ conversion tool that is part of the XSQ Tools package.

10

Is library index correction supported?

An XSQ library index reassignment tool is provided in the XSQ Tools package at the following site:

http://solidsoftwaretools.com/gf/project/xsq

11

Why does the converter not work on my variant of Linux?

The table in Overview describes that conversion *to* XSQ requires CentOS 4.7 or later, and conversion *from* XSQ requires CentOS 5.5 or later. Other versions of Linux are not supported.

12

What are the reserved quality values (QVs) in the XSQ file?

From the XSQ File Format Specification:

> For the QV, the range of the 6 bits is between 0 and 63. Valid quality values will be from 3 to 62, represented as PHRED scores [QV = -10*log10P(error)]. 0 and 1 are reserved for future usage; 2 indicates a missing quality value (not trimming); 63 refers to an uninformative or missing call.

13

What values should I use for libraryInsertSizeMinimum and libraryInsertSizeMaximum when they are not known during the conversion of mate-pair and paired-end files?

Any values will work as long as `libraryInsertSizeMinimum` is less than or equal to `libraryInsertSizeMaximum` and both values are nonnegative.

14

How can I collect statistics on an XSQ file?

Not supported currently. An XSQ Statistics utility to extract the number of reads by file, barcode, tag, etc., in addition to library type and read length, is under consideration.

15

How can I tell if my XSQ file is valid or corrupt?

Not supported currently. An XSQ validation utility is under consideration.

16

I have a CSFASTA file to convert to XSQ format, but I do not have the QUAL file. Can I create the XSQ file?

When converting CSFASTA to XSQ, if the QUAL file is missing or not found, all reads are marked for filtering in the output XSQ file. As a result, the reads are ignored (not processed) when used as input into LifeScope™ Software analyses.

The workaround is to create a dummy QUAL file to use as input with the `convertToXSQ.sh` script. Create a QUAL file with the same bead ids (and in the same order) as the CSFASTA file. Use QV values at least high enough to pass default thresholds.

17

I see an XSQ script in the LifeScope™ Software install tree. Can I use that script to convert my files?

No. The XSQ script installed with LifeScope™ Software is for internal use only. Command line users must download the XSQ Tools package to convert CSFASTA files to the XSQ format. (The LifeScope™ GUI interface also handles CSFASTA conversion.)

18

Where does the tool get the indexing kit info? Is it store in the xsq file or does it come from the XSQ_Tool package?

This information is distributed within XSQ Tools. See `XSQ_Tools/xsqreindexer/conf` for details.

19

If a wrong kit was used to assign the original xsq file, can the tool reassign it to a different, correct kit? E.g. from a 16-index kit to a 96-index kit?

No. Kit reassignment is not supported here because the index calls are of a different length (5 vs. 10 calls).

20

The 5500 ICS v1.0 and v1.1 only uses 4 colors of the 5-color 16 barcode kit. Can the index reassignment tool use all 5 colors for reassignment?

Currently the index reassignment tool is using the same underlying library, so for now the same issue is present for the 16 index library. This is not a problem with the 96 index libraries.

21

How can I tell if an XSQ file was generated from and ECC run?

As this is a property of each tag and may be different between tags, the IsBasePresent attribute is most informative. See `/RunMetadata/TagDetails/<TagName>/IsBasePresent` in the XSQ file.

Release Notes

September 21, 2011

reIndex.sh now supports relative file path in the command line arguments

reIndex.sh now checks for same Input and output file and throws an error if they are same

reIndex.sh now allows shell variable export (PATH -> LD_LIBRARY_PATH)

reIndex.sh: RunMetadata->LibraryDetails->ECC corrected to RunMetadata->LibraryDetails->ERCC

reIndex.sh: TT02300: Support for missing panels and missing library names

convertFromXSQ.sh: TT02231: Support for libraries/indexes with no beads

convertFromXSQ.sh: TT02290: Support for panels with no reads

convertToXSQ: TT02476: NumFragmentsPassed attribute is now updated properly

convertFromXSQ.sh: TT02931: Call/QV conversion from XSQ to CSFASTA/QUAL was incorrect for missing data (QV=63) when call was not 3. Restated, XSQ call/QV in (252, 253, 254, 255) should be converted to a csfasta call of '.' and a quality value of -1, which was happening properly for (255); (252, 253, and 254) were being converted to a call of A and a quality value of 62 incorrectly.

January 6, 2012

reIndex.sh: updated hdf libraries to 2.8 to resolve crashes with large datasets

reIndex.sh: the size of the new xsq will be equivalent to the size of the input xsq file, fixing a bug that was doubling the size in the newly created file.

convertFromXSQ.sh: TT02390: Removed PrimerFColorCallQV as required dataset to support corrected colorspace calls.

convertFromXSQ.sh, convertToXSQ.sh, reIndex.sh: Fixed CentOS6 behavior. (Changed bash operator from '<' to '-lt' for accurate glibc comparison).