Olink® Explore Preprocessing

Technical Information

1345, 4.7.1, 2024-03-07

Table of contents

1. Introduction and overview

Preprocessing is a necessary step in all Olink® Explore analyses. During preprocessing, the Next Generation Sequencing (NGS) output from the sequencing instrument is converted to counts files containing the number of reads for each Olink® sequence and a run metadata file containing additional information which is necessary for importing the counts files into the analysis software.

1.1 Conversion of NGS data to counts files

The conversion of NGS data to counts files is performed by ngs2counts, a redesigned successor to bcl2counts. The instrument, flowcell and NGS data format are automatically detected.

Currently ngs2counts can process NGS data from the instruments and flowcells listed below.

- Illumina NovaSeq X/X Plus 1.5B, 10B
- Illumina NovaSeq 6000 SP, S1, S2, S4 with XP kit, S4 with standard workflow (i.e. without XP kit)
- Illumina NextSeq 1000/2000 P2, P3
- Illumina NextSeq 500/550 High, Mid
- MGI DNBSEQ-T7 High-throughput

Please contact Olink support for information about processing NGS data from additional instruments and flowcells.

2. Installation of the preprocessing software

Installation is performed according to the chosen alternative, 1 or 2 as described below, and requires Linux system administration knowledge. Detailed installation instructions can be found at the end of the document:

- *7.1 explore-listener*
- *7.3 ngs2counts*

2.1 Alternative 1: Full installation of the complete explore-listener package

The recommended procedure is to perform the full preprocessing installation which includes the companion software explore-listener, which "listens" to the NGS run output folder, for example `/mnt/data/NovaSeqRuns`. Every time an instrument creates a new sequencing sub-folder, for example `230315_VH00391_52_AAAMN3HM5`, this folder is detected by explore-listener and ngs2counts is started automatically when the sequencing is complete.

Important note:
For Illumina instruments explore-listener will only start ngs2counts for NGS runs where an Olink custom sequencing recipe has been used. For example, if the recipe is called `Recipe_P3_v01_R1_8c_20dk_8c_22dk_8c_Noindexes_Noread2` the run will be skipped by explore-listener. To produce counts file for this recipe the user needs to run ngs2counts manually.

2.1.1 Running ngs2counts manually after installation of the explore-listener package

Sometimes it may be necessary to run ngs2counts manually. It can be done running the following template command:

```
docker run -v <FULL_PATH_TO_NGS_RUNS>:/ngsRunFolder --entrypoint ngs2counts --env
OLINK_LOG_LEVEL=debug localhost/explore-listener-image:<VERSION> /ngsRunFolder/<RUN_
FOLDER_NAME>
```

where FULL_PATH_TO_NGS_RUNS must be replaced with the folder containing instrument run folders and RUN_FOLDER_NAME with the name of the run folder and VERSION with the installed version, for example:

```
docker run -v /mnt/data/NovaSeqRuns:/ngsRunFolder --entrypoint ngs2counts --env
OLINK_LOG_LEVEL=debug localhost/explore-listener-image:v4.2.0 /ngsRunFolder/230315_
VH00391_52_AAAMN3HM5
```

If podman is used then the command is:

```
podman run -v <FULL_PATH_TO_NGS_RUNS>:/ngsRunFolder --entrypoint ngs2counts --env
OLINK_LOG_LEVEL=debug localhost/explore-listener-image:<VERSION> /ngsRunFolder/<RUN_
FOLDER_NAME>
```

It is required to give the NGS run folder as input argument to ngs2counts. There are additional options that can be passed to ngs2counts, for example for overriding the automatic index plate detection, see section *4. Input to ngs2counts.*

2.2 Alternative 2: Install only ngs2counts

This alternative may be interesting for users who have their own data pipeline, who cannot allow explore-listener to write lock and done files to the NGS run folder and/or who do not want to use Docker or podman. Place a single file, the ngs2counts binary, in a folder on the system PATH and ensure it is executable. Then either run ngs2counts manually from the command-line for every NGS run, or integrate it into a data pipeline.

3. Automatic detection of product, index plate and panel/block

The product, i.e. Explore HT or Explore 3072, the index plates and the panels/blocks are by default automatically detected by ngs2counts.

The detection is performed independently for each pooled sequencing library. The concept of sequencing library is related, but not equivalent, to the flowcell lane concept. For example, an S4 flowcell with XP kit will process four libraries in total - one in each of the four lanes, so for an S4 flowcell with XP kit there is a one-to-one mapping between lane and library. In contrast, an S4 flowcell with standard workflow will process a single large library.

The automatic detection relies on the internal control assays. It can handle low or no reads for many of the internal controls, but a technical error that results in all internal controls having close to no reads will make automatic detection difficult or impossible.

First the product - Explore 3072 or Explore HT - is detected by analyzing reads for internal control assays, which are different between the two products. The software will then detect index plates and panels (Explore 3072) or blocks (Explore HT) according to certain standard run scenarios. The standard scenario for Explore 3072 is to use either 1 or 4 index plate(s) per pooled sequencing library. The set of standard scenarios for Explore HT is more complex, and not listed here. If the lab has successfully applied a non-standard run scenario, then ngs2counts will output some extra counts files, containing close to no counts, in order to match one of the standard scenarios. The extra counts files can be ignored.

Panel detection is a step that is unique to Explore 3072, since the assay barcodes are shared between the panels. When the panel detection is successful, the panel type is included in the file name of the counts file. If the detection is unsuccessful the panel type in the file name will be set to `NA`. More details about file names are given in section *5.3 Output files from ngs2counts*.

4. Input to ngs2counts

The NGS run folder is a required input argument to ngs2counts. The following additional options are provided by ngs2counts when run from the command-line, irrespective of installation alternative:

1. `-o, --output-dir <DIR>`: Set output directory. If unset, the output will be written to the NGS run folder.
2. `--run-units <RUN_UNITS>`: Select index plates and panels/blocks manually, overriding the automatic detection. The alternatives are `all-explore3072` and `all-explore-ht`. Please note that this option shall not be used in normal scenarios, it is for trouble-shooting purposes only.
3. `--library <NUMBER>`: Select a single library to output data from, for example the library sequenced in lane 2 of an S4 flowcell with XP-kit. Please note that this option shall not be used in normal scenarios, it is for trouble-shooting purposes only.
4. `--num-libraries <NUMBER>`: Deprecated, use the library-mapping option instead! Optionally set the number of libraries that was run. This option can only be used on instruments and flow cell where the lanes are separable. Please note that this option shall not be used in normal scenarios, it is for non-standard run setups only.
5. `--library-mapping <LIB_MAPPINGS>`: Optionally give instructions on how a run with separable lanes should be mapped into libraries. Can also be used to rescue runs where one or more lanes has/have failed. A comma seperated list of mappings, where a map is given by `LIBRARY=[LANE+...+LANE]`.
   NOTE: A lane can only belong to one library! Example: `1=[1+2+4]`, maps lanes 1,2, and 4 to library 1; `1=[1+3],2=[2+4],3=[5+6+7+8]`, maps lanes 1 and 3 to library 1, lanes 2 and 4 to library 2, and lanes 5, 6, 7, and 8 to library 3.
6. `-h, --help`: Print the help text, which includes a list of required and optional input.
7. `-V, --version`: Print the version of ngs2counts.

5. Preprocessing output

5.1 Lock files written by explore-listener to NGS folders

When processing an NGS sequencing sub-folder, for example to `230315_VH00391_52_AAAMN3HM5`, an `explore-listener.lock` file will be written to the folder to hinder concurrent execution on the same folder from other installations. When the processing is done a `explore-listener.done` file will be written and the lock-file will be removed. The done-file marks the NGS folder as already processed so that it will not be processed again. If the program should fail for any reason, the user can trigger reprocessing of the NGS folder by manually removing the lock-file (or done-file) from the NGS folder.

5.2 Output folder for ngs2counts

The output generated by ngs2counts is by default written to the NGS sequencing sub-folder, for example to `230315_VH00391_52_AAAMN3HM5`, where all the instrument-generated output is already stored. Alternatively, it is possible to configure explore-listener to create a new subfolder with the same name, for example `230315_VH00391_52_AAAMN3HM5`, in a separate main output directory which is chosen during installation of explore-listener. When running ngs2counts manually, the option `--output-dir` can be used to set the output folder.

In addition to this there is an option `--add-experiment-name` for explore-listener that will append the experiment name to the folder created in the outputs folder, for example `230315_VH00391_52_AAAMN3HM5_My_Experiment`.

5.3 Output files from ngs2counts

Counts files are named according to the pattern

```
counts_{SEQUENCING_DATE}_{FLOWCELL_SIDE}{FLOWCELL_ID}_L{LIBRARY_NUMBER}_
P{INDEX_PLATE}_{ASSAY_LIBRARY}.csv
```

where LIBRARY_NUMBER indicates the order number of the pooled sequencing library, where INDEX_PLATE can be A or B for Explore HT and 1-4 for Explore 3072, and where ASSAY_LIBRARY can be Block 1-8 for Explore HT and one of the eight panels or NA for Explore 3072.

Example for Explore HT: `counts_2023-01-01_BHXYZDEF_L1_PA_Block_2.csv`
and for Olink Explore 3072: `counts_2023-01-01_AHXYZABC_L2_P1_ONCII.csv`

The file `run_metadata.json` contains metadata for the run, and is necessary for importing counts files into Olink NPX Explore or Olink Explore CLI.

5.4 Logging

Log messages from ngs2counts are written to stdout and stderr, which means they are printed on the screen unless redirected to a file. The level of verbosity is controlled via the environment variable `OLINK_LOG_LEVEL`. When running ngs2counts manually from the explore-listener container, it is recommended to set `--env OLINK_LOG_LEVEL=debug`.

When running ngs2counts as a standalone binary, the logging level is configured when running using `OLINK_LOG_LEVEL=debug ngs2counts <RUN_DIR> <OPTIONS> &> <LOG_FILE>`, for example `OLINK_LOG_LEVEL=debug ngs2counts 230315_VH00391_52_AAAMN3HM5 &> messages.txt` The `&> <LOG_FILE>` part pipes both the stdout and stderr output of the program to `LOG_FILE`. If the `OLINK_LOG_LEVEL` environment variable is not set only warnings and errors will be logged. Possible values are `error`, `warn`, `info`, and `debug`.

By default, explore-listener will redirect all ngs2counts log messages to a log file in the same folder as the counts files. It is named after the run folder according to the pattern <run_dir_name>_log.txt, for example `230315_VH00391_52_AAAMN3HM5_log.txt`. This log file must always be included when contacting support regarding a specific run.

6. Differences between bcl2counts and ngs2counts

This section is intended for customers who are upgrading the preprocessing software from bcl2counts to ngs2counts.

6.1 Counts files

The names of Explore 3072 counts files are different between bcl2counts and ngs2counts. Please see section *5.3 Output files from ngs2counts* for details.

6.2 ngs2counts is not compatible with MyData

The counts files and run metadata generated by ngs2counts cannot be uploaded to MyData, nor will support for upload to MyData be implemented in the future.

6.3 ngs2counts is only compatible with Olink NPX Explore v 1.8.0 or higher

The counts files and run metadata generated by ngs2counts cannot be imported into NPX Explore version 1.7.1 or older.

6.4 ngs2counts does not require Python

With ngs2counts it is no longer necessary to install Python nor any Python packages.

6.5 Logging

Log messages from ngs2counts are written to stdout and stderr. The user can choose to redirect stdout/stderr to a file. For bcl2counts the log messages were by default written to a file, and the user would have to set option `--stdout` to get the log messages to stdout.

Unlike for bcl2counts, the verbosity of ngs2counts log messages cannot be controlled via an option. The verbosity is controlled using the `OLINK_LOG_LEVEL` environment variable, which can be set to `error`, `warn`, `info`, or `debug`. The default is `warn`.

6.6 Output in case of severe technical errors or non-Olink data

The number of counts files generated by bcl2counts by default is controlled via the flowcell type. This means that when bcl2counts is run on non-Olink data, or a sequencing run where severe technical errors have prevented any Olink sequences to be read by the instrument, bcl2counts will still generate counts files. In contrast, ngs2counts will only generate counts files for a library if at least one index plate and panel/block has counts for at least one of the internal control assays.

6.7 ngs2counts does not support index plates with version 0

Index plates with version 0, more specifically Olink Explore Index plates with Part number 87005, Lot B04701 and EXPD 2023-05-31, are not supported by ngs2counts. Data from index plates version 0 can only be analyzed with bcl2counts.

7. Appendix - Installation instructions

7.1 explore-listener

7.1.1 Requirements

- Linux server with 32 GiB memory
- Read access to instrument runs from the Linux server
- Docker or podman

Note:
when using podman instead of docker all commands that are not specifically written for podman can be run by replacing `docker` with `podman`.

7.1.2 Steps

Before the installation, make sure that you have acquired the explore-listener image tarball.

1. Load the image from the tarball in either Docker or podman.

```
docker load -i explore-listener.tar.gz
```

```
gunzip -c explore-listener.tar.gz | podman load
```

2. Find the image name or the image ID.

```
docker image lD
```

3. Run the image as a container. Replace the FULL_PATH_TO_NGS_RUNS with the full path to the directory where the NGS runs are stored. Example `/mnt/data/NovaSeqRuns`

```
docker run --name localhost/explore-listener:<VERSION> --restart unless-stopped -d -v
<FULL_PATH_TO_NGS_RUNS>:/runs <image name or image ID> --runs-dir /runs
```

The listener will start parsing the input runs directly and write count file(s) and a run metadata file per NGS run.

To verify that the container is running, run:

```
docker ps
```

This command will show a table in the terminal. In this table there should be a row with the IMAGE explore-listener-image which were created recently (if the run command was run recently) and have the status Up. In this table you can also see the id and name of the container which can be useful when retrieving logs for that container and also when stopping and removing it.

To add a custom name to a container use the `--name` name flag followed by name for the container. It is important that the flag written before the image name in the command, i.e before `explore-listener-image` in the docker run command.

7.2 Restart on error

To make sure the container restarts if it would crash use the `--restart unless-stopped` flag. There are also other options for this flag, see docker documentation or podman documentation.

7.2.1 Stop the container

To stop the container run:

```
docker stop <container name or container ID>
```

7.2.2 Update the installation

When receiving a new tarball file make sure the previous container is stopped and then repeat the normal installation. If you want the new container to have the same name as the previous one, then you also need to remove the old container. To remove a container you first need to stop it and then remove it using:

```
docker rm <container name or container ID>
```

Note:

When removing a container the associated logs will be lost. More on logging later.

7.2.3 Specify output folder

The container can be run to write the output to a specified output folder instead of writing to the ngs folder it processes. To do this add another `-V` flag to the docker run command like this:

```
docker run --name explore-listener --restart unless-stopped -d -v <FULL_PATH_TO_NGS_
RUNS>:/runs -v <FULL_PATH_TO_OUTPUT_FOLDER>:/outputs localhost/explore-listener-
image:<VERSION> --runs-dir /runs -o /outputs
```

Where FULL_PATH_TO_OUTPUT_FOLDER is a full path to a folder. It will be created if it does not exist for docker. For podman it will need to be created before running the command. VERSION is the installed version of the explore-listener container such as `v4.2.0`.

7.2.4 Specify status files folder

In order to store information on whether a run folder has been processed, the container stores some small status files in the run folder. In environments where the owner of the container process hasn't got write access to the run folder, it's possible to store these status files in a seperate folder. To do that add another `-v` flag and a parameter to the docker run command like this:

```
docker run --name explore-listener --restart unless-stopped -d -v <FULL_PATH_TO_
NGS_RUNS>:/runs -v <FULL_PATH_TO_OUTPUT_FOLDER>:/outputs -v <FULL_PATH_TO_STATUS_
FOLDER>:/status localhost/explore-listener-image:<VERSION> --runs-dir /runs --run-
status-dir /status -o /outputs
```

Where FULL_PATH_TO_STATUS_FOLDER is a full path to a folder where the status files will be stored. It will be created if it does not exist for docker. For podman it will need to be created before running the command.

7.2.5 Logging

To get the logs of a container you can run:

```
docker logs <container name or container ID>
```

The date interval for logs can be set by using the `--since` and `--until` flags. Those flags show logs since/before timestamp (e.g. 2013-01-02T13:23:37Z) or relative (e.g. 42m for 42 minutes). The `--tail` flag can also be used to show a number of lines from the end of the log.

The logs can also be written to journalctl. By adding `--log-driver=journald` to the docker run command the logs can be accessed by `journald CONTAINER_NAME=<container name>`. This enables getting a continuous log between installations. This requires systemd.

In addition a log file will be written per run in the output folder.

7.2.6 NGS run folders

When processing NGS run folders a `explore-listener.lock` file will be written to the folder to hinder concurrent execution on the same folder from other installations. When the processing is done a `explore-listener.done` file will be written and the lock-file will be removed. The done-file marks the NGS folder as already processed so that it will not be processed again. If the program should fail for any reason the NGS folder can be reprocessed by removing the lock-file (or done-file) manually.

7.2.7 Verifying container setup

A program called fake-run-data is also included. This can be used to create a run containing faked data to use as input to test that the container setup is correct. To run this binary first run:

```
chmod +x fake-run-data
```

Then run:

```
OLINK_LOG_LEVEL=debug ./fake-run-data --instrument nextseq2000 --flowcell P2
--product-type 3K <output dir>
```

This will create the output dir folder (if it does not exist) and add one run to it. The full path to that folder can then be used in the docker run command to test that the container setup works. To verify the output after testing the installation run the fake-run-data program again on the folder that contains the counts file(s) but this time with the `--validate` argument.

```
OLINK_LOG_LEVEL=debug ./fake-run-data --instrument nextseq2000 --flowcell P2
--product-type 3K --validate <run dir>
```

7.3 ngs2counts

7.3.1 Requirements

- Linux server with 32 GiB memory
- Read access to instrument runs from the Linux server

7.3.2 Steps

1. Install the binary ngs2counts on path. Example: `/usr/local/bin/ngs2counts`.
2. Ensure the file is executable

7.3.3 Logging

The program logs to stdout and stderr. The level of verbosity is controlled via the environment variable `OLINK_LOG_LEVEL`. Possible values are `error`, `warn`, `info`, and `debug`, where `warn` is the defaul

7.4 User instructions

For user instrucitons, refer to *4. Input to ngs2counts*.

8. Revision history

Version	Date	Description
4.7.1	2024-03-07	*1.1* and *2.1* Instruments and flowcells updated.
4.5.0	2024-02-05	*4* bullet 4–5 updated. *5.4* updated.
4.4.0	2023-12-15	*7.2.3* added.
4.3.1	2023-11-15	*4* bullet 4 updated. Changed version number to match the software version.
6.0	2023-10-31	*2* updated. *7* added.
5.0	2023-10-12	*4* bullet 4 added *5.2* supplementary information added. Podman added RUST_LOG replaced by OLINK_LOG_LEVEL
4.0	2023-08-14	Overall changes
3.0	2023-04-05	New

www.olink.com

1345, 4.7.1, 2024-03-07