NPX™ Map CLI

Technical Information

1591, 1.0.2, 2025-02-10

Table of contents

1. Introduction

NPX™ Map CLI is a command-line interface (cli) for the *Olink® Explore HT* and the *Olink® Explore 3072* and Reveal. The application is capable of performing quality control (QC), normalization and CV computations on NGS data and exporting the results on several supported formats.

Fore Research Use Only. Not for use in diagnostic procedures.

1.1 Olink® Explore Software terminology and documentation

For detailed descriptions of Explore software terminology and output files, please refer to the user guide for [*NPX™ Explore HT & 3072 software*, which shares the same Explore software library for normalization, QC and output file generation.

For detailed descriptions of software terminology and output files, please refer to the user guide for *NPX™ Map software*, which shares the same software library for normalization, QC and output file generation.

Terminology	Description
Pre-processing	Conversion of next generation sequencing (NGS) output to counts per Olink Explore index-barcode sequence.
Plate layout file	A csv-file containing sample ID and sample type per well of one 96-well plate.
Run unit	The data found in a single counts file, corresponding to one 96-well plate and one Explore HT block or one Explore3072/Reveal panel.

1.1.1 Output files

Output file	Description
NPX file	A parquet-file with one record per combination of sample/control/external control and assay/internal control.
Extended NPX file	A parquet-file containing the same columns as the NPX file with additional columns.
CLI Data Export file	A parquet-file containing the same columns as the extended NPX file with additional columns and records for empty wells and excluded run units

1.2 Requirements

The program has the following system requirements:

1. A reasonably new Linux operating system. Supported distributions are LTS version of Ubuntu (ex. 22.04 and 24.04) and RHEL (8, 9). Most other modern Linux distributions should work as well, but are not tested by Olink.
2. *Docker*/*Podman*, or *Apptainer*, depending on which container image is used.

2. Installation

The supported ways of running the CLI are with Docker/Podman or Apptainer.

2.1 Apptainer

2.1.1 Installing the container image

Extract the zip file containing the CLI Apptainer .sif image.

The NPX Map CLI container is invoked through the `apptainer run` command:

```
apptainer run npx-map-cli.sif info
```

2.2 Docker or podman

2.2.1 Installing the container image

Extract the zip file containing the CLI Docker image and import it:

```
docker load npx-map-cli.tar

podman load -i npx-map-cli.tar
```

The NPX Map CLI container is invoked through the `docker run` command:

```
docker run --rm ghcr.io/olink-proteomics/npx-map-cli:<VERSION> info
```

For podman run the following command:

```
podman run --rm ghcr.io/olink-proteomics/npx-map-cli:<VERSION> info
```

where VERSION is the downloaded version of the npx-map-cli such as `1.0.0`.

2.2.2 Running the container while bind mounting the host file system

This does not apply to Apptainer workflows.

The following command processes a run definition on json format `runs.json` and outputs a parquet export file in the same directory. Please note that relative/absolute file paths referenced in the json file must also exist within the bind mounted directory for them to be visible to the container. The user argument is necessary to keep the file ownership of the generated parquet file to the user executing the container. For Podman change `docker` to `podman`.

Please see *2.4 Panel data archive* about configuring the necessary panel data archive file. This does not apply to Apptainer workflows.

```
docker run --rm \
    --mount type=bind,source=”${PWD}”,target=/data \
    -u $(id -u ${USER}):$(id -g ${USER}) \
    ghcr.io/olink-proteomics/npx-map-cli:<VERSION> runs -i /data/runs.json -o /data/
-p /data/NPXMap_PanelDataArchive_<VERSION>.dat
```

2.3 Migration guide for explore-cli users

Users coming from `explore-cli` 1.x and 2.x will need to adjust their CLI arguments to be able to use NPX Map CLI.

The differences are as follows:

- The `create` and `export` verbs are replaced with the `runs` verb that takes a json file describing NGS runs and outputs a NPX parquet file.
- The runs input json file is a subset of the old project json format, old project json files that worked in explore-cli **also work** in NPX Map CLI.
- Since there is not dedicated export verb the user must choose the desired NPX output file directly in the `runs` verb.
- NPX Map CLI manages panel data differently than explore-cli, see *2.4 Panel data archive* about Panel Data Archive files for more details.

Please note that counts folders generated with NPX Map CLI are **not compatible** for processing with explore-cli 1.x and 2.x.

Counts generated with `ngs2counts` can still be processed with explore-cli.

2.4 Panel data archive

NPX™ Map CLI requires panel reference data in the form of a Panel Data Archive file in order to perform QC and normalization on NGS data.

The file is distributed together with the NPX™ Map CLI software.

The archive file can be made accessible to npx map in one of two ways:

- Adding the `-p` or `--panel-data-archive` argument to the CLI verbs requiring it. The argument must specify either an absolute or relative path to the panel data archive file to use.
- Setting the `NPX_MAP_PANEL_DATA_ARCHIVE` environment variable before running the CLI software. The value must be a valid absolute file path pointing to the panel data archive file to use.

The `info` verb can be used to verify that a panel data archive file is correctly configured and display information about its contents.

2.5 Versioning

Panel data archives are versioned in the same manner as the software.

A specific version of the software can utilize panel data archive that is of the same version or higher allowing for running with newer panel data than was initially provided with the software.

Version strings only differing in the third digit (patch version) are treated as the same version for compatibility checks, i.e. version `1.0.1` is treated as equal to version `1.0.0`.

The `minimumSoftwareVersion` specifies how old the software can be and still utilize the panel data archive in question.

2.6 Logging

The program writes logs to `STDOUT` that can be redirected to a file if need be. The log level can be controlled with the environment variable `OLINK_LOG_LEVEL` and has the following values where `warn` is default:

- trace
- debug
- info
- warn
- error

Should an unrecognized environment value be set the default log level `warn` will be used.

The log level can be passed to the container with the respective argument for each container runtime:

2.6.1 Docker

```
docker run --rm -e OLINK_LOG_LEVEL=info ghcr.io/olink-proteomics/npx-map-
cli:<VERSION> --help
```

2.6.2 Apptainer

```
apptainer run --env OLINK_LOG_LEVEL=info npx-map-cli.sif info
```

2.7 Verbs

The Olink Explore CLI is divided into several smaller functions called verbs. Each verb is responsible for performing a specific action in the NPX Map workflow. The available verbs are the following:

- info
- fastq
- ultima
- ultima standard
- runs
- schema
- readme

2.7.1 Verb: info

Displays relevant information about the software.

Short option	Long option	Required	Description
-p	--panel-data-archive	no	Path to the Panel data archive file to use if not configuring it with the environment variable.

Outputs a json formatted string with the following information:

Field	Description
versions.softwareVersion	The version of the CLI generating this information.
versions.normalizationAndQcSpecification	The version of the specification for normalization and QC calculations.
versions.outputFileFormat	The version of the data output format and specification.
versions.researchUseLabel	Research use only statement.
panelData.version	The version of the panel data archive.
panelData.minimumSoftwareVersion	The minimum software version that the configured panel data archive supports.
panelData.products	Information of the available panels in the configured panel data archive.

Example 1

```
npx-map-cli info
```

2.7.2 Verb: fastq

Perform preprocessing of one or more FASTQ files.

Short option	Long option	Required	Description
-i	--input	**yes**	Input json file specifying what to process.
-o	--output	no	Path of folder in which to save the counts folder, defaults to current working directory if not set.

This is used for converting FASTQ files to a count folder. The FASTQ file(s) needs to contain data from one library only which means that the tool generating the FASTQ file needs to split instrument data into libraries if more than one library has been run.

Input is a json file containing the parameters necessary to map the FASTQ file(s) to counts. This command can also process FASTQ file(s) over stdin. Relative path are resolved against the directory of the json input file.

Example json input format 1

```
{
    “inputType”: “FastqFiles”,
    “fastqFiles”: [
          “relative/path/to/file_1.fastq.gz”,
          “/absolute/path/to/file_2.fastq.gz”
    ],
    “productType”: “ExploreHT”,
    “instrumentId”: “ABC123”,
    “instrumentType”: “DNBSEQT7”,
    “experimentName”: “My experiment”,
    “uniqueRunIdentifier”: “ABCD1234”
}
```

Example json input format 2

```
{
    “inputType”: “FastqFolder”,
    “fastqFolder”: “/path/to/fastq/files”,
    “productType”: “ExploreHT”,
    “instrumentId”: “ABC123”,
“instrumentType”: “AVITI”,
“experimentName”: “My experiment”,
“uniqueRunIdentifier”: “ABCD1234”
}
```

Please see provided json schema files (schemas/fastq.json) for details on the json input specification.

Example 1

```
npx-map-cli fastq -i input.json -o /path/to/my_counts_folders/
```

2.7.3 Verb: Ultima

Operations for processing ultima data.

2.7.4 Verb: Ultima standard

Perform preprocessing of an Ultima run folder.

Short option	Long option	Required	Description
-i	--input	**yes**	Input Ultima run folder to process.
-o	--output	no	Path of folder in which to save the counts folder, defaults to the Ultima run folder if not set.
	--instrument -id		Optional instrument id to annotate the run with, will be present in generated run_metadata.json
	--ignore -xml	no	Override xml check. The xml file is usually read to gain information about if the run is a HT run or not. This flag overwrites that check.

This is used for converting a Ultima run folder with a trimmer histogram file to a counts folder.

The output is stored in the same folder as the input folder unless --output is given.

Output

Counts files are named according to the pattern:

```
counts_{DATE}_{BARCODE_LABEL}_L1_P{INDEX_PLATE}_{ASSAY_LIBRARY}.csv
```

where DATE is the date that the counts file was created and the BARCODE_LABEL is the barcode label present in the trimmer file name in the sub folder in the Ultima run folder. INDEX_PLATE can be A or B and ASSAY_LIBRARY can be Block 1-8.

The field `runIdentifier` in the `run_metadata.json` will consist of the folder name of the provided run folder and the field `experimentName` will be the BARCODE_LABEL.

Ultima folder structure and file formats

The run folder is expected to contain a `{RUN_ID}_LibraryInfo.xml` where RUN_ID is the id of the run. It also needs to contain a sub folder containing the trimmer histogram csv file. If the program is run with the `--override-xml` flag, there is no need to have the xml file.

The trimmer histogram file is expected to be located in a sub folder in the Ultima run folder and have the file name `{BARCODE_LABEL}-FBC_name-RBC_name-sample_index_name_hist.csv` where BARCODE_LABEL should have the format Zdddd (Z and then 4 digits).

The csv file is expected to be separated by `,` and contain four columns named `FBC_name`, `RBC_name`, `sample_index_name` and `count`. Where the `FBC_name` column contains the detected forward barcodes separated by a `+`.

The `RBC_name` column contains the detected reverse barcodes separated by a `+`. The `sample_index_name` column contains the detected sample index, and the `count` column contains the number of occurrences of the combination in the row.

Example 1

```
npx-map-cli ultima standard -i /path/to/ultima_runFolder/ -o /path/to/my_counts_
folders/
```

2.7.5 Verb: runs

Perform batch processing of runs.

Short option	Long option	Required	Description
-i	--input	**yes**	Input file for describing the runs.
-o	--output	no	Directory to create output in, defaults to current working directory if not set.
-t	--export-type		The type of NPX file to export in parquet format.
-p	--panel-data-archive	no	Path to the Panel data archive file to use if not configuring it with the environment variable.

Create an NPX file from an input definition json file. The input file contains one or more run folders produced by the pre-processing such as the fastq command, ultima command or from the ngs2counts program, one or more plate layouts to be used and which data analysis reference ids that are to be used for each panel or block.

Possible values for the --export-type (-t) option are: NPX, ExtendedNPX, CLIDataExport. Default is: CLIDataExport.

Please note that NPX values and all QC results are independent between plates also when Intensity Normalization is chosen for the project. Adding, removing or replacing a plate in a project does not affect NPX nor QC results of any of the other plates. One and only one export file column depends on all plates together: InterCV.

This column is included in Extended NPX file and CLI Data Export File.

Example json input format 1

```
{
    “projectName”: “TestProject-CSAS3_EXPL”,
    “productType”: “ExploreHT”,
    “normalization”: “Intensity”,
    “sampleMatrix”: “Blood plasma”,
    “annotations”: {
          “key1”: “value1”,
          “key2”: “value2”
    },
    “selectedDataAnalysisRefIds”: [
          “D10001”,
          “D20001”,
          “D30001”,
          “D40001”,
          “D50001”,
          “D60001”,
          “D70001”,
          “D80001”
    ],
    “plateLayouts”: [
    {
          “path”: “plate_layouts/plate1.csv”
          “plateId”: “plate1”
    }
    ],
    “runs”: [
            {
                  “path”: “230405_A00915_0850_AHYMWNDSX3”,
                  “runUnits”: [
                        {
                              “plateLayout”: “plate1”,
                              “libraryNumber”: 1,
                              “indexPlate”: “A”,
                              “panel”: “Block_1”,
                              “included”: true
                        },
                        {
                              “plateLayout”: “plate1”,
                              “libraryNumber”: 1,
                              “indexPlate”: “A”,
                              “panel”: “Block_2”,
                              “included”: true
                        }
                  ]
            }
      ]
}
```

Please note that when index plate “C” has been used in the lab, index plate “A” and “B” shall still be set in the json file. The field “indexPlate”: “A” shall be set for index 1 to 96 and “indexPlate”: “B” shall be set for index 97 to 192.

Please see provided json schema files (schemas/runs.json) for details on the json input specification.

If no export type is provided the program exports the default CLI Data Export file in *Apache Parquet* format.

For documentation of NPX file and Extended NPX file, please refer to the user guide for *NPX™ Map*.

Example 1

```
{
px-map-cli runs -i input.json -o /path/to/my_npx_files/ --export-type NPX
}
```

2.7.6 Verb: schema

Outputs the json schema for different command input types

Outputs the json schema for different command input types

Example 1

```
{
npx-map-cli schema fastq
}
```

2.7.7 Verb: readme

Prints out the README for NPX™ Map CLI in markdown format.

Short option	Long option	Required	Description
-v	--verb	no	The specific verb to display README text for, leave out to generate the entire README.

3. Appendix

3.1 Project data file (Apache Parquet)

The parquet file contains multiple columns and here follows a table explaining each column.

- The **Name** column presents the column names in the parquet file.
- The **Scope** column present which level the data occurs. For example if the scope is project, then that value applies to the whole project. If the scope value instead is datapoint then the value is unique for each datapoint.
- The **Type** column presents the data type of the values in the column.
- The **Example** column presents an example of what the data in the column could look like.
- The **Description** column provides a small description of the column.

Name	Scope	Type	Example	Comment
SampleID	sample	string	subject-123	Identifier from sample_id column of plate layout file. Combination SampleId-OlinkId must be unique within set of included run units in project, otherwise invalid input. Restrictions on sample_id in the plate layout file: At most 100 characters, no comma and no semicolon.
SampleType	sample	string	SAMPLE_CONTROL	Sample type as given in plate layout file. Possible values are : PLATE_CONTROL, NEGATIVE_CONTROL, SAMPLE_ CONTROL, SAMPLE and EMPTY (CLI export only)
WellID	sample	string	A1, B3	Well on 96-plate as given in plate layout file.
PlateID	plate	string	SS123456	Name of plate layout file without extension.
DataAnalysisRefID	run unit	string	D10001	Data analysis reference id entered by user, deciding which reference values to use for Quality control metrics.
OlinkID	assay	string	OID20790	Olink assay identifier.
UniProt	assay	string	Q86VW0	Uniprot ID
Assay	assay	string	SESTD1	Assay name
AssayType	assay	string	assay	Assay type of datapoint. Possible values are **assay, amp_ctrl, inc_ctrl, ext_ctrl**.
Panel	run unit	string	Explore_HT, Cardiometabolic, Inflammation_II	For product Explore HT panel is always Explore_HT. For Product Explore3072 possible values are: Oncology, Neurology, Cardiometabolic, Inflammation, Oncology_II, Neurology_II, Cardiometabolic_II, Inflammation_II

Name	Scope	Type	Example	Comment
Block	Block	string	1, A	Dilution block. For product ExploreHT possible values are 1, 2, 3, 4, 5, 6, 7 and 8. For product Explore3073 possible values are A, B, C, D
Count	data-point	int	2641	0: NA. Datapoint is for assay that did not pass Olink's batch release quality control > 0: Same integer as in corresponding record in count column of counts file.
ExtNPX	data-point	double	-1.94701	The 2-logarithm of the ratio between the Count value for the datapoint and the Count value for the Extension control assay in the same block and for the same sample. Value is NaN for assays that did not meet Olink's quality control criteria for batch release, and for datapoints which have failed on a sample-block and/or plate-block level.
NPX	data-point	double	1.735509	NPX for project chosen normalization. Bimodal assays are always plate control normalized. Value is NaN for assays that did not meet Olink's quality control criteria for batch release, and for datapoints which have failed on a sample-block and/or plate-block level.
Normalization	data-point	string	Plate control	Plate control or Intensity or EXCLUDED (only assays that did not meet Olink's quality control criteria for batch release can have EXCLUDED and is always EXCLUDED). Always plate control for bimodal assays. Remaining assays are normalized with project-selected normalization.

Name	Scope	Type	Example	Comment
PCNormalizedNPX	data-point	double	1.735509	Plate control (PC) normalized NPX. This column is independent of the normalization type chosen for the project. The values in this column will be identical to the values in the NPX column if Plate Control normalization has been chosen for the project. The values in this column will be different from the values in the NPX column if Intensity normalization has been chosen for the project. Value is NaN for assays that did not meet Olink's quality control criteria for batch release, and for datapoints which have failed on a sample-block and/or plate-block level.
AssayQC	data-point	string	PASS	This column gives a string representation of the AssayQCWarn column. There are three possible values, each with a 1-to-1 mapping to the integer values in AssayQCWarn in the Extended NPX file. NA: This datapoint is for an assay that did not meet Olink's quality control criteria for batch release or for an internal control assay or for an assay on a plate without any passed negative controls that can be used to perform assay QC. Value is NA if and only if AssayQCWarn==0. PASS: This datapoint does not have an assay warning on a plate level. Value is PASS if and only if AssayQCWarn==1. WARN: This datapoint has an assay warning on a plate level. Value is WARN if and only if AssayQCWarn==2

Name	Scope	Type	Example	Comment
SampleQC	data-point	string	PASS	This column gives a string representation of the overall QC status for this sample in this block. It has a strict mathematical mapping to the combination of SampleBlockQCWarn, SampleBlockQCFail and BlockQCFail in the Extended NPX file. There are four possible values: NA, PASS, WARN, FAIL. In each block, each unique SampleID will have the same value in this column for all assays that did meet Olink's quality control criteria for batch release. Assays that did not meet these criteria will have value NA. In other words, each SampleID can have at most one non-NA SampleQC value in each block. NA: This datapoint represents an assay that did not meet Olink's quality control criteria for batch release and does not have a QC status. Value is NA if and only if SampleBlockQCWarn==0 and SampleBlockQCFail==0 and BlockQCFail==0. PASS: The sample has passed all QC checks in the sample dimension in this QC unit. Value is PASS if and only if SampleBlockQCFail == 1 and BlockQCFail == 1 and SampleBlockQCWarn < 2. WARN: The sample has a warning on the sample - block level. Value is WARN if and only if SampleBlockQCFail == 1 and BlockQCFail == 1 and SampleBlockQCWarn >= 2 FAIL: The sample has failed, either on the sample - block level or for the entire block and plate. Value is FAIL if and only if SampleBlockQCFail > 1 or BlockQCFail > 1, where the "or" is inclusive. Note: SampleBlockQCWarn is not needed to determine if the value is FAIL. SampleBlockQCWarn can be 0, 1 or > 1 for datapoints that have failed.
SoftwareVersion	run unit	string	1.0.0	The semantic version X.Y.Z of the software that was used to produce the data.
SoftwareName	run unit	string	NPX Map	Software name. Possible values are NPX Map or NPX Map CLI.
PanelDataArchiveVersion	run unit	string	1.0.0	The version X.Y.Z of the Panel Data Archive that was used for the QC.

Name	Scope	Type	Example	Comment
PreProcessingVersion	run unit	string	1.0.0	The version of the preprocessing software as indicated in run_metadata.json for the run unit.
PreProcessingSoftware	run unit	string	NPX Map	name of pre-processing software. Possible values: NPX Map CLI, NPX Map, ngs2counts
InstrumentType	run unit	string	Illumina NextSeq 2000	NGS instrument type read from pre-processing run_metadata.json Possible values: Element Biosciences AVITI MGI Tech DNBSEQ T7 Illumina NextSeq 550 Illumina NextSeq 1000 Illumina NextSeq 2000 Illumina NovaSeq 6000 Illumina NovaSeq X Illumina NovaSeq X Plus Ultima Genomics UG100
IntraCV	assay-plate	double	0.101	Intra CV For chosen normalization, except always plate control for bimodal assays. Value is NaN for assays that did not meet Olink's quality control criteria for batch release, for assays not included in CV calculations, and for datapoints which have failed on a sample-block and/or plate-block level.
InterCV	assay-project	double	0.201	Inter CV For chosen normalization, except always plate control for bimodal assays. Value is NaN for assays that did not meet Olink's quality control criteria for batch release, for assays not included in CV calculations, and for datapoints which have failed on a sample-block and/or plate-block level.
SampleBlockQCWarn	data-point	int	0	0: NA. This datapoint is not for type SAMPLE, or this datapoint is for an assay that did not meet Olink's quality control criteria for batch release. 1: PASS. This datapoint is for type SAMPLE and there is no warning on the sample-block level. >1: WARN. This datapoint is for type SAMPLE and there is a warning on the sample-block level. For details on the interpretation of values greater than 1, see *Documentation of integer representation*.

Name	Scope	Type	Example	Comment
SampleBlockQCFail	data-point	int	1	0: NA. This datapoint is for an assay that did not meet Olink's quality control criteria for batch release. 1: PASS. This datapoint is not failed on a sample-block level. >1: FAIL. This datapoint is failed on a sample-block level. For details on the interpretation of values greater than 1, see *Documentation of integer representation*.
BlockQCFail	data-point	int	1	0: NA. This datapoint is for an assay that did not meet Olink's quality control criteria for batch release. 1: PASS. This datapoint is not failed on a plate-block level. >1: FAIL. This datapoint is failed on a plate-block level. For details on the interpretation of values greater than 1, see *Documentation of integer representation*.
AssayQCWarn	data-point	int	1	0: NA. This datapoint is for an assay that did not meet Olink's quality control criteria for batch release or internal control assay or for an assay on a plate without any passed negative controls that can be used to perform assay QC. 1: PASS. This datapoint does not have an assay warning on a plate level. 2: WARN. This datapoint has an assay warning on a plate level. For more details see [*Documentation of integer representation*]
RunID	run unit	string	8ca76722-d1fd-4a4a-a296-d77415675651	A uuid. Unique run identifier created by preprocessing software and written to run_metadata.json
RunUnitId	run unit	string	8ca76722-d1fd-4a4a-a296-d77415675651	A uuid. Unique run unit (count file) identifier created by preprocessing software and written to run_metadata.json

Name	Scope	Type	Example	Comment
ExperimentName	run unit	string	LJ111-1111_ SS123456_NEU_INF	Written to run_metadata.json by preprocessing software. For runs preprocessed with ngs2counts the string is copied from Illumina output RunParameters.xml. For runs preprocessed with NPX Map and verb ultima the string is the name of the input run folder. For runs preprocessed with NPX Map and verb fastq the string is an optional input with default NA.
RunIdentifier	run unit	string	HHCYVDRXY	Written to run_metadata.json by preprocessing software. For runs preprocessed with ngs2counts the string is copied from Illumina output RunInfo.xml, field "Flowcell". For runs preprocessed with NPX Map and verb ultima the string is an optional input. For runs preprocessed with NPX Map and verb fastq the string is a required input.
InstrumentID	run unit	string	A01234	Written to run_metadata.json by preprocessing software. For runs preprocessed with ngs2counts the string is copied from Illumina output RunInfo. xml. For runs preprocessed with NPX Map the string is an input with default NA.
LibraryNumber	run unit	int	1	Read from run_metadata. "Library" is short for "pooled sequencing library". For runs preprocessed with ngs2counts: The default for a NovaSeq6000 S4 flowcell with Xp workflow is to treat each lane as a separate library, giving a one-to-one mapping between lane and library, but the default can be overridden via option --library-mapping. A NovaSeq S4 flowcell with standard workflow can only sequence a single library, since the lanes are not separated. For runs preprocessed with NPX Map: LibraryNumber will always be 1, since only one pooled sequencing library can be preprocessed at a time.
IndexPlate	run unit	string	A	Sample index plate, i.e. range of sample indices, for run unit. Can be 'A' or 'B' in Explore HT and '1', '2', '3', '4' for Explore 3072

Name	Scope	Type	Example	Comment
SampleIndexVersion	run unit	int	2	Sample index version used in counts file. Version is 2 or 5 for all Explore HT index plates and 1 for Explore3072 index plates
MatchedCounts	library	long	354689590	The total number of matched counts for the run unit (count file).
Reads	library	long	638337024	Will be 0 for Ultima run data preprocessed with NPX Map verb ultima. For any other type of run data this is the total number of reads in the library with number LibraryNumber in run RunID.
Included	run unit	bool	TRUE	Indicates whether run unit (count file) is included in or excluded from the project.
PreProcessingRunTimestamp	run unit	DateTime	2001-01-01 02:00:00	Input read from pre-processing run_ metadata.json, ISO DateTime
AssayCategory	assay	int	0	0 represents an internal control or regular biomarker assay that passed batch release quality criteria. 1 represents an assay that failed batch release quality control criteria for the batch corresponding to the selected Data Analysis Reference Id.
AssaySystematicEffect	data-point	int	0	0: NA. This datapoint has not been subject to checks for systematic effects. 1: PASS. This datapoint has no flag for any assay-level systematic effects on this plate. >1: FLAG. This datapoint has a flag for one or more assay-level systematic effects on this plate. For more details see *Documentation of systematic effect integer representation*.
BlockSystematicEffect	data-point	int	0	The BlockSystematicEffect flags when a sufficiently high fraction of individual assays flag for the same type of systematic effect. 0: NA. This datapoint has not been subject to checks for systematic effects. 1: PASS. This datapoint has no flag for any block-level systematic effects on this plate. >1: FLAG. This datapoint has a flag for one or more block-level systematic effects on this plate. For more details see *Documentation of systematic effect integer representation*.

3.2 Documentation of detailed integer representation of QC status

Extended NPX file columns SampleBlockQCWarn, SampleBlockQCFail, BlockQCFail and AssayQCWarn together give the QC status of each datapoint on a sample, block and assay level. The values are encoded using the same principle as the FLAG value in Sequence Alignment/Map format (SAM) files used to represent aligned RNA and DNA sequences.

In each of the four columns, the value is the sum of the numeric representations of all the individual QC check results for that column and datapoint. The numeric representation unambiguously encodes the complete QC status for the datapoint. The detailed interpretation of each value can be obtained either programmatically, by bit-masking using the binary representation of each QC check, or manually, using the integer representations.

SampleBlockQCWarn QC check results	Integer representation	Binary representation
NA: The checks do not apply for this datapoint.	0	00000
PASS: All the checks pass for this datapoint.	1	00001
Warning due to low incubation control assay count	2	00010
Warning due to low amplification control assay count	4	00100
Warning due to low extension control assay count	8	01000
Warning due to low incubation control ratio in SAMPLE	16	10000

Important note: A datapoint for type SAMPLE which fails due to low incubation control count will not have a warning due to low incubation control count.

The same rule applies for amplification control assay and extension control assay. The failure will be indicated in the SampleBlockQCFail column.

SampleBlockQCWarn Example 1: A regular sample has a warning due to both low extension control assay count and to low amplification control count.

The numeric representation is the sum of the individual representations for those two warnings: 8+4=12, so the value in the SampleBlockQCWarn column is 12.

SampleBlockQCWarn Example 2: A Plate Control is not subject to the checks that can lead to a sample warning. The numeric representation for "The checks do not apply for this datapoint" is 0, so the value in the SampleBlockQCWarn column is 0.

BlockQCFail QC check results	Integer representation	Binary representation
NA: The checks do not apply for this datapoint	0	000
PASS: All the checks pass for this datapoint.	1	001
Fail due to too few passed Negative Controls	2	010
Fail due to too few passed Plate Controls	4	100

AssayQCWarn QC check results	Integer representation	Binary representation
NA: The checks do not apply for this datapoint.	0	00
PASS: All the checks pass for this datapoint.	1	01
Warning due to unexpected signal in Negative Controls	2	10

3.3 Documentation of detailed integer representation of systematic effects

Columns AssaySystematicEffect and BlockSystematicEffect together give the complete systematic effect information presented in the user interface of NPX Map.

The flags are intended as a starting point for further investigation and must not be interpreted as reports of confirmed systematic effects.

The systematic effects flags are first checked on a plate-assay level. An effect can flag on a plate-assay level without there being a plate-block level flag.

Multiple patterns can flag for the same assay, however some combinations are not possible, as explained below.

1. All full plate patterns are checked before individual column or row effects, and if any full plate pattern flags then the row and column effect checks are not performed.
2. The full plate Column Gradient effect is checked before the full plate Four Column Pattern 1-3, and if the Column Gradient effect flags then the Four Column Pattern effect checks are not performed.

When an individual row or column effect is detected, the output does not include information about which row or column is affected. The four different full plate Diagonal Gradient effects differ by shifts of the diagonal.

The integer value of AssaySystematicEffect is the sum of the numeric representations of all the individual check results for the assay and plate.

AssaySystematicEffect systematic effect check results	Integer representation	Binary representation
NA: The checks do not apply for this datapoint. Checks do not apply for internal control assays, nor for assays that did not pass batch release QC criteria, nor for assays where NPX data in unavailable	0	0000000000000
PASS: The checks apply for this assay and datapoint, and no effect is flagging for this assay and plate	2^0	0000000000001
Full plate Column Gradient	2^1	0000000000010
Full plate Row Gradient	2^2	0000000000100
Full plate Diagonal Gradient 1	2^3	0000000001000
Full plate Diagonal Gradient 2	2^4	0000000010000
Full plate Diagonal Gradient 3	2^5	0000000100000
Full plate Diagonal Gradient 4	2^6	0000001000000
Full plate Alternating Column Pattern	2^7	0000010000000
Full plate Four Column Pattern 1 flags but not the Column Gradient	2^8	0000100000000
Full plate Four Column Pattern 2 flags but not the Column Gradient	2^9	0001000000000
Full plate Four Column Pattern 3 flags but not the Column Gradient	2^10	0010000000000
Individual Column Effect flags but none of the full plate effects flag	2^11	0100000000000
Individual Row Effect flags bug none of the full plate effects flag	2^12	1000000000000

The block-level systematic effect is a summary of the assay-level flags. The block-level effects flag when a sufficiently high fraction of individual assays in that block flag for the same type of systematic effect. The integer value of BlockSystematicEffect is the sum of the numeric representations of all the individual check results for the block and plate.

BlockSystematicEffect systematic effect check results	Integer representation	Binary representation
0: NA. This datapoint is not subject to checks for systematic effects	0	0000000000000
1: PASS. For this plate, no assay-level effect flags for a significant fraction of assays in this block	2^0	0000000000001
Full plate Column Gradient flags for a significant fraction of assays	2^1	0000000000010
Full plate Row Gradient flags for a significant fraction of assays	2^2	0000000000100
Full plate Diagonal Gradient 1 flags for a significant fraction of assays	2^3	0000000001000
Full plate Diagonal Gradient 2 flags for a significant fraction of assays	2^4	0000000010000
Full plate Diagonal Gradient 3 flags for a significant fraction of assays	2^5	0000000100000
Full plate Diagonal Gradient 4 flags for a significant fraction of assays	2^6	0000001000000
Full plate Alternating Column Pattern flags for a significant fraction of assays	2^7	0000010000000
Full plate Four Column Pattern 1 flags for a significant fraction of assays	2^8	0000100000000
Full plate Four Column Pattern 2 flags for a significant fraction of assays	2^9	0001000000000
Full plate Four Column Pattern 3 flags for a significant fraction of assays	2^10	0010000000000
Individual Column Effect flags for a significant fraction of assays	2^11	0100000000000
Individual Row Effect flags for a significant fraction of assays	2^12	1000000000000

4. Revision history

Version	Date	Description
1.0.2	2025-02-10	Software update does not affect the manual.
1.0.1	2025-01-27	New