

# FAST5 format de-mystified

Hasindu Gamaarachchi, Hiruna Samarakoon, Sasha P. Jenner, James M. Ferguson, Timothy G. Amos, Jillian M. Hammond, Hassaan Saadat, Martin A. Smith, Sri Parameswaran, Ira W. Deveson

Kinghorn Centre for Clinical Genomics, Garvan Institute of Medical Research, Sydney, NSW, Australia.

## PREAMBLE

FAST5 files are Hierarchical Data Format 5 (HDF5) files with a specific schema defined by Oxford Nanopore Technologies (ONT) for storing raw current-signal data generated from ONT devices. We have compiled the following material to help researchers to understand FAST5 files, and consulted ONT on various definitions within their format. This document should not be interpreted as a definitive specification document from the developers of FAST5, although it is the most detailed description of FAST5 format that we are aware of.

There are two FAST5 types: single-FAST5 and multi-FAST5 (first appearing around September 2018). A multi-FAST5 file contains a batch of reads in a single file whereas a single-FAST5 file contains just a single read per file. Single-FAST5 format is no longer used by ONT. In this document, FAST5 will always refer to multi-FAST5 unless otherwise stated.

To read FAST5 files we use the HDF5 library and HDF5 tools [1].

## BASICS

A FAST5 (HDF5) file is like a file system. Just as there are multiple levels of **directories** and **files** in a file system, a FAST5 (HDF5) file contains **groups** and **datasets**, respectively. The term **HDF5 objects** is an umbrella term for both groups and datasets. HDF5 objects can optionally contain **attributes**, which are key-value pairs. HDF5 related terms are defined below with examples.

### Groups

HDF5 groups (and links<sup>1</sup>) organise HDF5 objects. Every HDF5 file contains a root group that can contain other groups or links to other HDF5 objects<sup>2</sup>. Working with groups and group members (HDF5 objects) is similar in many ways to working with directories and files in UNIX. As with UNIX directories and files, objects in an HDF5 file are often described by giving their full (or absolute) path names.

- `/` signifies the root group.
- `/foo` signifies a member of the root group called *foo*.
- `/foo/zoo` signifies a member of the group *foo*, which in turn is a member of the root group.

---

<sup>1</sup> Links are like directory/file paths in a file system. Links can be absolute, relative or even symbolic.

<sup>2</sup> Other objects can, in theory, be in the same FAST5 file or in a different FAST5 file.

## Datasets

HDF5 datasets organise and contain the actual data values [2]. A dataset consists of metadata (datatype, datasize, compression technique, etc) that describes the data, in addition to the data itself. In any read within a FAST5 file, two datasets are found; the *Raw* group contains the raw current-signal the *Analyses* group contains the FASTQ data (only if live base-calling was enabled) [1].

## Attributes

Attributes can optionally be associated with HDF5 objects. The attributes have two parts: a name and a value. Attributes are accessed by opening the object that they are attached to; hence, they are not independent objects. Typically an attribute is small in size and contains details about the object that it is attached to.

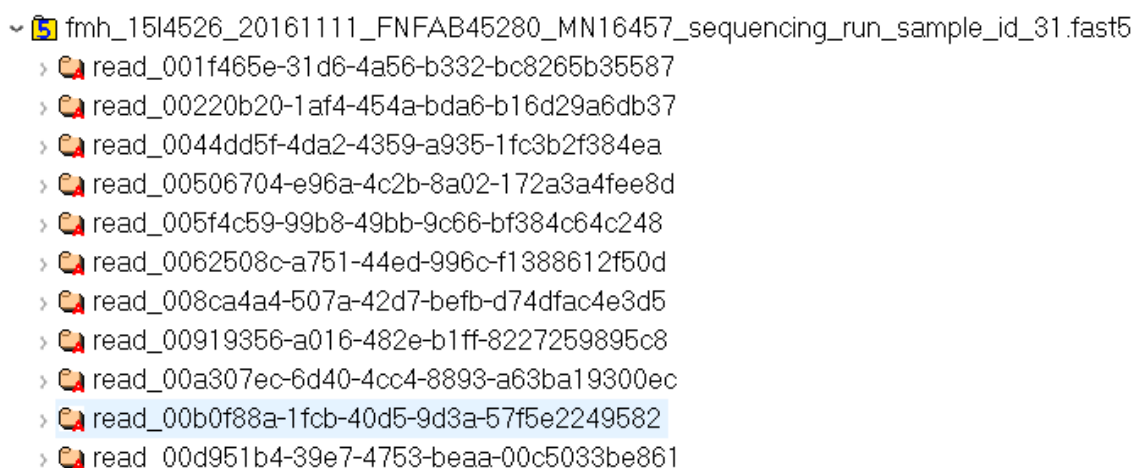
Attributes look similar to HDF5 datasets in that they have metadata such as data type and dataspace. However, unlike HDF5 datasets, HDF5 attributes do not support partial I/O operations and cannot be compressed or extended [2].

## HDF5 dataspace

The HDF5 *dataspace* must be defined prior to defining an HDF5 dataset or an attribute. The dataspace defines some metadata such as the size and shape of the dataset or attribute raw data (i.e., the number of dimensions and the size of each dimension of the multidimensional array in which the raw data is represented) [2].

## HIERARCHY OF A MULTI-FAST5 FILE

The root group (e.g. /fmh\_15... in the snapshot below) contains a group for each read that is named as “read\_” followed by the read identifier (e.g. read\_001f4... in the snapshot below):

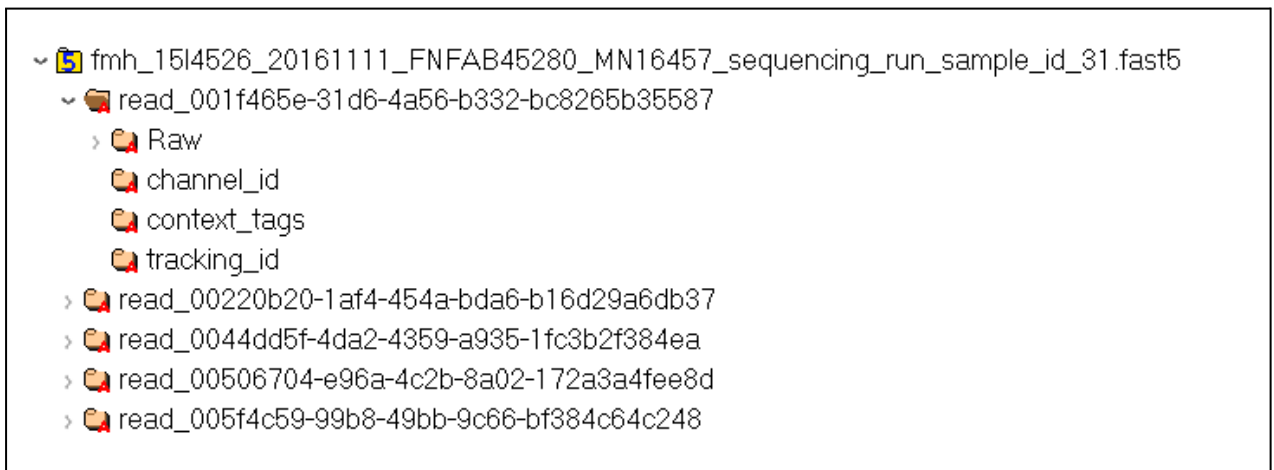


```
fmh_15l4526_20161111_FNFAB45280_MN16457_sequencing_run_sample_id_31.fast5
├── read_001f465e-31d6-4a56-b332-bc8265b35587
├── read_00220b20-1af4-454a-bda6-b16d29a6db37
├── read_0044dd5f-4da2-4359-a935-1fc3b2f384ea
├── read_00506704-e96a-4c2b-8a02-172a3a4fee8d
├── read_005f4c59-99b8-49bb-9c66-bf384c64c248
├── read_0062508c-a751-44ed-996c-f1388612f50d
├── read_008ca4a4-507a-42d7-befb-d74dfac4e3d5
├── read_00919356-a016-482e-b1ff-8227259895c8
├── read_00a307ec-6d40-4cc4-8893-a63ba19300ec
├── read_00b0f88a-1fcb-40d5-9d3a-57f5e2249582
└── read_00d951b4-39e7-4753-beaa-00c5033be861
```

Under each read group, there are the following groups:

- Raw
- channel\_id
- context\_tags
- tracking\_id
- Analyses

Note that the *Analyses* group is only available in base-called FAST5 files. The groups found in a FAST5 file that has not been base-called are shown in the snapshot below:



As the name suggests, *Raw* contains the raw signal (raw data acquisition values) and associated metadata. *channel\_id* contains (but is not limited to) parameters useful for converting the raw signal into pico-ampere values. *context\_tags* and *tracking\_id* contain global information that are common to the sequencing run. More information on these groups is provided below.

The *Analyses* group is for storing data resultant from various downstream analyses, such as base-calling. For instance, if the Guppy base-caller is run with the option to output base-called FAST5 files, those output FAST5 files will contain this *Analyses* group. *Analyses* groups can be used by custom software (e.g. Tombo) for storing data from additional downstream analyses.

In the following subsections we provide detailed descriptions of groups mentioned above, except the *Analyses* group. Since we are concerned with the raw signal data, basecalled data is not in the scope of this document.

### **Root\_group**

Root group has two attributes *file\_type* (note that *file\_type* is only available in multi-fast5 from version 2.2 onwards) and *file\_version*. Note that we do not make any assumptions about file-structure based on FAST5 version numbers, because we have observed some inconsistencies across different files of the same version, and we would discourage users from doing so.

#### **1. file\_type**

*Example value: multi-read*

*Data type: String*

#### **2. file\_version**

*Example value: 2.2*

*Data type: String*

## Read

A read group has two attributes *run\_id* and *pore\_type* (note that *pore\_type* is not available in multi-fast5 v2.0).

### 1. run\_id

The value of this attribute is constant across all the read groups.

*Example value:* fe697f519ab04ba540bc4fe93f7cbd86669f38ca

*Data type:* String

### 2. pore\_type

In existing FAST5 versions, this value is empty. This attribute may be used in the future to distinguish different pore types within a single flow cell.

*Example value:* <not set>

*Data type:* String

## Raw

*Raw* group contains one dataset and seven attributes. The dataset is the raw signal, which is a series of 16-bit integers (HDF5 Datatype = *H5T\_STD\_I16LE*). These are the integer values directly coming from the data acquisition process (analog to digital converter). This raw signal can be converted into pico Ampere (pA) values using attributes available in the *channel\_id* group (explained later).

The seven attributes from the *Raw* group are listed below with a description of each attribute, example values and the data type. Understanding these descriptions require a brief understanding of an ONT flow cell. A flow cell has multiple channels allowing multiple DNA/RNA strands to be sequenced in parallel. For instance, a MinION flow cell has 512 channels and thus can sequence 512 strands in parallel. Each channel contains one or more wells<sup>3</sup>. For instance, a MinION flow cell has 4 wells per channel. The wells within a channel are connected to a multiplexer (MUX), a switch that controls which of the four wells in the channel is controlled and read out by the circuits. Please refer to reference [3] or [4] for more information about channels and multiplexers.

Note that some of the information below is extracted from ONT document [1].

### 1. read\_number

A unique number within each channel counted upwards from zero [1]. Note that not all reads generated are “strand” reads, but only strand reads are written to the final fast5 file, so some read numbers may be absent.

*Example value:* 17981

*Data type:* 32-bit signed integer

### 2. read\_id

A unique identifier for the read. This is a Universally unique identifier (UUID) version 4 and should be unique for any read from any device.

*Example value:* 00592138-f120-4ab5-9916-c5567adb8e29

*Data type:* String

---

<sup>3</sup> Each well should ideally contain one pore

### 3. **start\_time**

The start time of the read. The unit for *start\_time* is 'number of signal samples', so *start\_time* has to be divided by sampling rate ( $\text{Read\_xxxx}/\text{channel\_id}/\text{sampling\_rate}$ ) to get the start time in seconds (i.e. the time since the run was started).

*Example value:* 335845487

*Data type:* 64-bit unsigned integer

### 4. **duration**

The duration of the read. The unit for *duration* is also 'number of signal samples'.

*Example value:* 1467

*Data type:* 32-bit unsigned integer

### 5. **start\_mux**

The MUX setting<sup>4</sup> for the channel when the read began. Due to timing issues this can sometimes reflect what the MUX was just *before* the read began; this will only matter for reads that start immediately after a MUX change.

*Example value:* 4

*Data type:* 8-bit unsigned integer

### 6. **median\_before**

The estimated median current level immediately preceding the read. In most cases this can be used as an estimate of the open pore level<sup>5</sup>.

*Example value:* 238.78225708007812

*Data type:* 64-bit floating-point

### 7. **end\_reason**

This is a new attribute in FAST5 v2.2 onwards.

*Example value:* unblock\_mux\_change

*Data type:* 8-bit enum

```
ATTRIBUTE "end_reason" {
    DATATYPE H5T_ENUM {
        H5T_STD_U8LE;
        "unknown"      0;
        "partial"       1;
        "mux_change"    2;
        "unblock_mux_change" 3;
        "signal_positive" 4;
        "signal_negative" 5;
    }
}
```

## **Channel\_id**

This group has attributes that are relevant to the channel that sequenced a given read. The *channel\_id* group has the following attributes.

### 1. **channel\_number**

The channel number from which the read was acquired.

*Example value:* 504

---

<sup>4</sup> out of the wells in the channel, which well the mux is set to sequence

<sup>5</sup> open-pore state is when there is no strand inside the pore

*Data type: String*

## 2. digitisation

The digitisation is the number of quantisation levels in the Analog to Digital Converter (ADC). That is, if the ADC is 12 bit, digitisation is 4096 ( $2^{12}$ ).

*Example value: 8192.0*

*Data type: 64-bit floating-point*

## 3. offset

The ADC offset error. This value is added when converting the signal to pico ampere.

*Example value: 10.0*

*Data type: 64-bit floating-point*

## 4. range

The full scale measurement range in pico amperes.

*Example value: 1441.389892578125*

*Data type: 64-bit floating-point*

## 5. sampling\_rate

Sampling frequency of the ADC, i.e., the number of data points collected per second (in Hertz).

*Example value: 4000*

*Data type: 64-bit floating-point*

Of these attributes, *digitisation*, *offset* and *range* can be used to transform the raw signal in the *Raw* group (raw ADC values), to pico-ampere current values as follows:

$$\text{signal\_in\_pico\_ampere} = (\text{raw\_signal\_value} + \text{offset}) * \text{range} / \text{digitisation}$$

## Context\_tags

The *context\_tags* group has global attributes that describe the sequencing run. The attributes under the *context\_tags* group are listed below with short descriptions. The data type of the value of all the attributes listed under *context\_tags* group is String.

### 1. barcoding\_enabled

Indicates if barcode demultiplexing is enabled during live basecalling

*Example value: 0*

### 2. experiment\_duration\_set

Indicates the duration of the experiment selected when starting the sequencing run (in minutes)

*Example value: 4320*

### 3. experiment\_type

Indicates the type of the experiment, for instance, *genomic\_dna* or *rna*.

*Example value: genomic\_dna*

### 4. local\_basecalling

Indicates if live base calling is enabled or not (set to 1 or 0).

*Example value: 1*

### 5. package

This attribute relates to the Bream package

[[https://github.com/nanoporetech/minknow\\_lims\\_interface](https://github.com/nanoporetech/minknow_lims_interface)]

*Example value:* bream4

**6. Package\_version**

*Example value:* 6.0.7

**7. sample\_frequency**

Typically the same as the *sampling\_frequency* in the *channel\_id* group

*Example value:* 4000

**8. sequencing\_kit**

The sequencing kit selected by the user in the GUI, for instance, *sqk-lsk109* or *sqk-rna002*. [<https://store.nanoporetech.com/sample-prep.html>]

*Example value:* sqk-lsk109

There can be additional attributes such as *basecall\_config\_filename*, depending whether live basecalling was turned on/off when the sequencing run was started.

### **Tracking\_id**

The *tracking\_id* group has global attributes relevant to the sequencing run and the sequencing device. These are mostly for internal use by ONT, who have assisted us in providing the definitions below. The data type of the value of all the attributes listed under *tracking\_id* group is String.

**1. asic\_id**

Application Specific Integrated Circuit identifier (ASIC) of the flow cell (unique number of the chip). Enables tracking of batches of chips.

*Example value:* 213553007

**2. asic\_id\_eeprom**

Identifier of the ASIC's electrically erasable programmable read-only memory (EEPROM) of the flow cell.

*Example value:* 5309577

**3. asic\_temp**

The temperature in degrees celsius of the ASIC chip at the start of the sequencing run.

*Example value:* 28.867193

**4. asic\_version**

The version of ASIC being used.

*Example value:* 1A02D

**5. auto\_update**

Whether auto update in Minknow is enabled or not.

*Example value:* 0

**6. auto\_update\_source**

The link to the Minknow update source.

*Example value:* <https://mirror.oxfordnanoportal.com/software/MinKNOW/>

**7. bream\_is\_standard**

Bream is one of the software for controlling sequencing.

*Example value:* 0

**8. configuration\_version**

The version of the configuration system in MinKNOW including the experiment scripts.

*Example value: 4.0.13*

**9. device\_id**

The serial ID of the MinION or device position for GridION/PromethION.

Device position on GridION/PromethION refers to the ID of the bay (slot where the flowcell is put) on the device.

*Example value: X2*

**10. device\_type**

The device type, that is whether MinION, PromethION or GridION.

*Example value: gridion*

**11. distribution\_status**

Stable vs dev/alpha/beta status.

*Example value: stable*

**12. distribution\_version**

MinKNOW version.

*Example value: 20.06.9*

**13. exp\_script\_name**

The name of the experiment script run along with optional parameters passed to it, based on what kits are selected in MinKNOW for sequencing.

*Example value: sequencing/sequencing\_MIN106\_DNA:FLO-MIN106:SQK-LSK109*

**14. exp\_script\_purpose**

The 'purpose' of the experiment script. For example, whether the experiment was a real sequencing run or a simulation playback.

*Example value: sequencing\_run*

**15. exp\_start\_time**

Start time of sequencing run in ISO 8601 standard.

*Example value: 2020-09-08T01:23:21Z*

**16. flow\_cell\_id**

Unique ID for the flowcell, used by ONT to track flowcell metrics and warranty.

*Example value: FAN43349*

**17. flow\_cell\_product\_code**

The type of flowcell (product code of the flowcell and pore type). These will be different based on R9.4.1, R10.3, R9.5, PromethION, etc.

*Example value: FLO-MIN106*

**18. guppy\_version**

Guppy version being used by MinKNOW.

*Example value: 4.0.11+f1071ce*

**19. heatsink\_temp**

The temperature (in degrees celsius) of the heat sink on the ASIC at the start of the sequencing run.

*Example value: 33.996094*

**20. hostname**

The hostname of the computer/machine doing the sequencing run.



*Example value: GXB02243*

**21. installation\_type**

This is the MinKNOW installation type.

*Example value: nc*

**22. local\_firmware\_file**

*Example value: 1*

**23. operating\_system**

The operating system and version of the computer performing the sequencing run.

*Example value: ubuntu 16.04*

**24. protocol\_group\_id**

This is the unique ID given to the group of acquisition periods during a run, denoted by run\_id. Multiple acquisition periods can occur during a single “run”, depending on the protocol.

*Example value: GLFN180082*

**25. protocol\_run\_id**

This is a unique identifier for the experiment GROUP (just in case the name given by the user is not unique). This is the same for each run of the same experimental group.

*Example value: f2c69573-5fef-43b8-8d81-9cb20634aa7c*

**26. protocol\_start\_time**

The start time of the data acquisition periods for a *protocol\_group\_id*. Appeared in FAST5 2.3.

*Example value: 2021-08-26T15:34:52.186021+10:00*

**27. protocols\_version**

Allows MinKNOW to track various protocols for barcoding, kits, etc.

*Example value: 6.0.7*

**28. run\_id**

The unique run ID which will be different for each run (data acquisition period), even in the same experiment group. Whenever MINKNOW starts an experiment script for data acquisition, a new run\_id is generated.

*Example value: 07770780274b0e3703f00d969291b1a37a5a6be1*

**29. sample\_id**

Sample ID is the name given by the user for the sample.

*Example value: NA12878*

**30. usb\_config**

Information about the connection between the flowcell and the computer.

*Example value: GridX5\_fx3\_1.1.3\_ONT#MinION\_fpga\_1.1.1#bulk#Auto*

**31. version**

MinKNOW version.

*Example value: 4.0.3*

Note that the above list is not an exhaustive list. For instance, FAST5 files generated on the PromethION have additional attributes such as *hublett\_board\_id* and *satellite\_firmware\_version*.

## FAST5 VERSIONS & THEIR ATTRIBUTES

The following table shows the availability (and unavailability) of attributes in un-basecalled multi-FAST5 files for different file versions.

Green cells = attribute available.

Group	Attribute name	V2.0	V2.2	v2.3
/	file_type			
	file_version			
/read	run_id			
	pore_type			
/read/Raw	start_time			
	duration			
	read_number			
	start_mux			
	read_id			
	median_before			
	end_reason			
/read/channel_id	digitisation			
	offset			
	range			
	sampling_rate			
	channel_number			
/read/context_tags	barcoding_enabled			
	experiment_duration_set			
	experiment_type			
	local_basecalling			
	package			
	package_version			
	sample_frequency			
	sequencing_kit			
	experiment_kit			
	filename			
	user_filename_input			
/read/tracking_id	asic_id			
	asic_id_eeprom			
	asic_temp			
	asic_version			
	auto_update			
	auto_update_source			
	bream_core_version			

	bream_is_standard			
	bream_ont_version			
	bream_prod_version			
	bream_rnd_version			
	configuration_version			
	device_id			
	device_type			
	distribution_status			
	distribution_version			
	exp_script_name			
	exp_script_purpose			
	exp_start_time			
	flow_cell_id			
	flow_cell_product_code			
	guppy_version			
	heatsink_temp			
	host_product_code			
	host_product_serial_number			
	hostname			
	installation_type			
	local_firmware_file			
	operating_system			
	protocol_group_id			
	protocol_run_id			
	protocol_start_time			
	protocols_version			
	run_id			
	sample_id			
	usb_config			
	version			

## CONSTANT & VARIABLE ATTRIBUTES

Many FAST5 attributes are identical amongst all the reads within a single sequencing run (within multi-FAST5 files as well as amongst different multi-FAST5 files). For example, all reads from a given experiment will have the same *run\_id*. Some attributes are variable between different reads, even within a single multi-FAST5. For example, each read has a different *read\_id*. All the attributes in *contex\_tags* and *tracking\_id* are constant across all reads in a single sequencing run, whereas most of the attributes in *Raw* and *channel\_id* are variable between reads (with a few exceptions).

The variable attributes amongst all groups (except the *Analyses* group) are:

- duration
- end\_reason (not in version 0.6 but in 2.2)

- median\_before
- read\_id
- read\_number
- start\_mux
- start\_time
- channel\_number
- offset

Note that the dataset “Signal” obviously has variable data. All the other attributes are constant.

## ADVANCED INFORMATION

### Symbolic links

For a given FAST5 file, the values of the attributes belonging to the two groups, *context\_tags* and *tracking\_id* are the same for all the reads in that FAST5 file. Hence only the first read\_xxxx group has the actual attributes. The rest of the read\_xxxx groups maintain symbolic links [2] to the first read\_xxxx group. One can observe the linking structure of a FAST5 file using a utility program called *h5dump* developed by the HDF5 group.

The following is an example output for a FAST5 file where read\_000200a4\* is the first read group. As listed below the rest of the read groups’ *context\_tags* and *tracking\_id* attributes maintain links (symbolic links) to the *context\_tags* and *tracking\_id* attributes of the first read group respectively. This observation was valid for all the FAST5 files we have examined.

```
GROUP "context_tags" {
    HARDLINK "/read_000200a4-0347-4a49-b800-37ad7b4287c9/context_tags"
}
GROUP "tracking_id" {
    HARDLINK "/read_000200a4-0347-4a49-b800-37ad7b4287c9/tracking_id"
}
```

### ONT h5 validator

[Ont h5 validator](#) is a tool developed by ONT to check if a given FAST5 file complies with the FAST5 schema. This tool only considers a subset of the complete FAST5 schema to validate a file.

### Single-FAST5 format fields

The groups, attributes and some example values for a single-FAST5 file are provided below for the sake of completeness, despite no longer being in use.

<b>PreviousReadInfo</b> previous_read_id = cf435984-627d-450d-a81d-2a55c6060c80 previous_read_number = 80
<b>Read</b>

duration = 30695  
median\_before = 206.2032470703125  
read\_id = b3d473e9-34f0-4ad6-a030-61ba6ab458bc  
read\_number = 99  
start\_mux = 4  
start\_time = 318648

**channel\_id**

channel\_number = 707  
digitisation = 2048.0  
offset = -196.0  
range = 748.5801660113588  
sampling\_rate = 4000.0

**context\_tags**

experiment\_duration\_set = 3840  
experiment\_type = genomic\_dna  
fast5\_output\_fastq\_in\_hdf = 1  
fast5\_raw = 1  
fast5\_reads\_per\_folder = 4000  
fastq\_enabled = 1  
fastq\_reads\_per\_file = 4000  
filename = pct0028\_20181029\_0004a30b00232bec\_1\_e11\_h11\_sequencing\_run\_lxbab132606\_84140  
flowcell\_type = flo-pro002  
kit\_classification = none  
local\_basecalling = 1  
local\_bc\_comp\_model =  
local\_bc\_temp\_model = template\_r9.4\_450bps\_5mer\_raw.jsn  
sample\_frequency = 4000  
sequencing\_kit = sqk-lsk109  
user\_filename\_input = lxbab132606

**tracking\_id**

asic\_id = 0004A30B00232BEC  
asic\_id\_eeprom = 0004A30B00232BEC  
asic\_temp = 36.990513  
asic\_version = Unknown  
auto\_update = 0  
auto\_update\_source = <https://mirror.oxfordnanoportal.com/software/MinKNOW/>  
bream\_is\_standard = 0  
device\_id = 1-E11-H11  
device\_type = promethion  
exp\_script\_name = 59dfa94107ee2b6c0f4be0822482e7da35b4116a-da65898430ab8c4bfe54ba7064f0301390b76211  
exp\_script\_purpose = sequencing\_run  
exp\_start\_time = 2018-10-29T01:40:23Z  
flow\_cell\_id = PAD11989  
heatsink\_temp = 41.996017  
hostname = PCT0028  
hublett\_board\_id = 013220e36be4c748  
hublett\_firmware\_version = 2.0.5  
installation\_type = nc  
ip\_address =  
local\_firmware\_file = 1  
mac\_address =  
operating\_system = ubuntu 16.04  
protocol\_run\_id = e3b445eb-5626-48ef-acc2-b28bcc611009  
protocols\_version = 0.0.0.0  
run\_id = 855cdb4b269484b72699b681e539e090c4a50bbb  
sample\_id = LXBAB132606  
satellite\_board\_id = 0000000000000000  
satellite\_firmware\_version = 2.0.4

```
usb_config = firm_1.2.3_ware#rbt_4.5.6_rbt#ctrl#USB3  
version = 1.14.2
```

## REFERENCES

- [1] Oxford Nanopore Technologies. Read .fast5 files from the instrument. *technical\_documents*  
[https://community.nanoporetech.com/technical\\_documents/data-analysis/v/datd\\_5000\\_v1\\_rev\\_n\\_22aug2016/read-fast5-files-from-th](https://community.nanoporetech.com/technical_documents/data-analysis/v/datd_5000_v1_rev_n_22aug2016/read-fast5-files-from-th) (2016).
- [2] The HDF5 Group. HDF5 User's Guide: HDF5 Release 1.6.10. <https://support.hdfgroup.org/HDF5/doc1.6/UG/> (2009).
- [3] Lu, H., Giordano, F. & Ning, Z. Oxford Nanopore MinION Sequencing and Genome Assembly. *Genomics, Proteomics & Bioinformatics* vol. 14 265–279 (2016).
- [4] Lannoy, C. de, de Lannoy, C., de Ridder, D. & Risse, J. The long reads ahead: de novo genome assembly using the MinION. *F1000Research* vol. 6 1083 (2017).