Structure of JPK Archives.

JPK files are zipped archives, containing data and header files, arranged into a top level folder containing one header filer (named `header.properties`) and several subfolders. To my knowledge, there are 3 different types of archives:

- **simple force scan**, with default file extension `.jpk-force`. This type contains a single force spectroscopy recording.
- **nt force scan**, with default file extension `.jpk-nt-force`. This contains data recorded with JPK Instruments' NanoTracker system (optical tweezers).
- **force scan map**, default extension `.jpk-force-map`. This type contains a 'force map', a collection of simple force scans.

The following is a closer look at the archive structure. Common features are described first, followed by a list of features unique to the different archive types.

Common features of JPK archives

- There is a header file at the top level named `header.properties`. This file appears to contain parameters all subfolders have in common.
- There can be a folder named `shared-data`. It contains another header file with the same name as the top level header, by full path it should be `shared-data/header.properties`. It contains parameters that sometimes is referred to by header files in subfolders. It uses a system of indices to discriminate between blocks of parameters. This indices are then referred to by the header files in subfolders. **WARNING** I do not know if I covered all cases of links from local headers to the shared header. This might lead to errors in conversion from raw data to physical units. I list the cases that I implemented at the bottom under 'Links to shared header'.

Subfolders in simple force scans and nt force scans

These two archive types seem to have the same structure.

- There is a folder named `segments`. It contains subfolders with integers as nai subfolder stands for one segment of the recording.
    - Each subfolder contains a header file named `segment-header.properties`. The parameters of

latest

this header file are only valid for one single segment. Among the parameters is one specifying the type of segment: whether it's an extension, a retraction or a pause. It also contains information on what channels are present in this segment, and how to convert the raw data to physical data.

  - Each subfolder contains another folder named `channels`.
    - The `channels` folder contains pure data files, no headers or meta-data of any form. The files' names are identical to the channels' names, followed by the extension `.dat`. The data type is specified in the `segment-header.properties` file above. The data type has to be known to read the binary data in the files and convert it into a python data type. All data appears to be stored in C structs and can be converted using the python module `struct`'s function `unpack`.

Additional files in force scan maps

This archive type has some additional files at the top level:

- `thumbnail.jpg`, `data-image.force`. Both seem to be preview images, maybe with different file types for different systems, not sure.
- A folder named `index`. It contains subfolders with integers as names. Each of these folders holds the data of one pixel of the force map and parameters valid only for one pixel.
  - The contents of each of those folders are identical to that of a simple force scan. They contain another `header.properties` and a `segments` folder.

Links to shared header

General rule

The header files contain many parameters (one per line), they have the form of keywords separated by dots, followed by an equal sign and a value. Here is an example from a header file:

```
channel.ySignal1.data.num-points=16384
```

A parameter line can also contain a link to parameters in the shared header. They take the following form:

```
<keyword_1>.<keyword_2>...<keyword_n>.<link_label>-info.*=i
```

latest

where `i` is an integer.This links to parameters starting with `<link_label>-info.i` in the shared header, like so:

```
<link_label>-info.i.<keyword_a1>.<keyword_a2> = value_a
<link_label>-info.i.<keyword_b1> = value_b
```

and so on. These would then be considered in the local header as

```
<keyword_1>.<keyword_2>...<keyword_n>.<keyword_a1>.<keyword_a2> = value_a
<keyword_1>.<keyword_2>...<keyword_n>.<keyword_b1> = value_b
```

In other words, you leave out the label with the '-info' suffix and the integer and append the keywords and values that follow in the shared header.

Links from segments to shared header

The following are lists of where links to a shared header occured in the JPK archives that I used to test the `jpkfile` module.

1. For data conversion

For conversion of raw data to physical data, the headers contain a key called *conversion-set*, under which there can be multiple factors that need to be applied for the conversion. Instead of storing the *conversion-set* locally, it can be stored in the shared header. This seems to be the case if there is a keyword chain looking like this in the local header:

```
channel.<channel_label>.lcd-info.*=i
```

This integer links to a number in the shared header, occuring in the following style:

```
lcd-info.i
```

There can be multiple parameters starting with this keyword chain.

2. General segment information

latest

Some of the segment's general parameters can be stored in a shared header. There needs to be the following chain of keywords:

```
force-segment-header.force-segment-header-info.*=i
```

This links to parameters in the shared header starting with:

```
force-segment-header-info.i
```

latest