ndeftool documentation

Release 0.1.1

Stephen Tiedemann

Sep 06, 2020

Contents

The **ndeftool** is a command line utility for creating, modifying and printing NFC Data Exchange Format (NDEF) Records. It is licensed under the ISCL, hosted on GitHub and installable from PyPI.

```
$ pip install ndeftool
$ ndeftool --help
```

CHAPTER 1

NDEFTOOL

Create, modify or print NFC Data Exchange Format Records.

1.1 Synopsis

```
ndeftool [OPTIONS] COMMAND1 [ARGS]... [COMMAND2 [ARGS]...]...
```

1.2 Description

The **ndeftool** provides a number of commands to create, modify or print NDEF records. Commands can be chained together to successively extend or modify the list of records that are generated. Except for **save** and **print**, all records forwarded by the last command are send to standard output as binary NDEF message data. The **save** and **print** commands implicitly consume all records unless the `--keep` option is set. They also, unlike all other commands, do not start an empty record list but read from standard input when given as the first command (the same behavior can be achieved for the **load** command by setting the file path to `-`).

By default, NDEF records read from disk or standard input must be correctly formatted NDEF message data (for example, the first and the last record must have the message begin and end flag set). The `--relax` makes the decoder accept correctable data errors and the `--ignore` option will additionally skip records with uncorrectable errors.

1.3 Options

`--version`
: Show the version and exit.

`--relax`
: Ignore some errors when decoding.

--ignore
Ignore all errors when decoding.

--silent
Suppress all progress information.

--debug
Output debug progress information.

--help
Show this message and exit.

1.4 Commands

1.4.1 load

Load records or payloads from disk.

Synopsis

```
ndeftool load [OPTIONS] PATH
ndeftool l [OPTIONS] PATH
```

Description

The **load** command reads records or payloads from disk files or standard input. The files for reading are determined by the pattern specified by PATH, which in the simplest form is an existing file name. Other forms may include `*`, `?` and character ranges expressed by `[]`. A single `-` may be used to read from standard input. Note that patterns containg wildcards may need to be escaped to avoid shell expansion.

The default mode of operation is to load files containing NDEF records. In `--pack` mode the files are loaded into the payload of NDEF records with record type (NDEF Record TNF and TYPE) set to the mimetype discovered from the payload and record name (NDEF Record ID) set to the filename.

Options

-p, --pack
Pack files as payload into mimetype records.

--help
Show this message and exit.

Examples

Pack text from standard input and pack as record.

```
$ echo -n "Hello World" | ndeftool load --pack - print
NDEF Record TYPE 'text/plain' ID '<stdin>' PAYLOAD 11 byte '48656c6c6f20576f726c' ...␣
↪1 more
```

Read text from file and pack as record.

```
$ echo -n "Hello World" > /tmp/hello && ndeftool load --pack /tmp/hello print
NDEF Record TYPE 'text/plain' ID '/tmp/hello' PAYLOAD 11 byte '48656c6c6f20576f726c' .
↪.. 1 more
```

Read with path containing wildcard characters.

```
$ ndeftool load --pack 'i?d*.rst' print
NDEF Record TYPE 'text/plain' ID 'index.rst' PAYLOAD 627 byte '2e2e202d2a2d206d6f64' .
↪.. 617 more
```

Read and pack multiple files.

```
$ ndeftool load --pack '../se?up.*' print
NDEF Record TYPE 'text/plain' ID '../setup.cfg' PAYLOAD 141 byte '5b746f6f6c3a70797465
↪' ... 131 more
NDEF Record TYPE 'text/x-python' ID '../setup.py' PAYLOAD 2563 byte
↪'23212f7573722f62696e' ... 2553 more
```

1.4.2 save

Save records or payloads to disk.

Synopsis

```
ndeftool save [OPTIONS] PATH
ndeftool s [OPTIONS] PATH
```

Description

The **save** command writes the current records to disk. The records to write can be restricted to the subset selected with `--skip`, `--count`, `--head` and `--tail` applied in that order. The default mode is to save all selected records as one NDEF message into a single file given by PATH. In `--burst` mode each record is written as one NDEF message into a separate file under the directory given by PATH. The file names are three digit numbers created from the record index. In `--unpack` mode the payload of each record is written to a separate file under directory PATH with the file name set to the record name (NDEF Record ID). Records without name are not written unless `--unpack` and `--burst` are both set.

The **save** command does not replace existing files or directories unless this is requested with `--force`.

The **save** command consumes records from the internal message pipe. This can be prevented with `--keep`, all records are then forwarded to the next command or written to standard output. When **save** is the first command it creates the pipe by reading from standard input.

Options

`--skip` `N`
: Skip the first N records.

`--count` `N`
: Skip the first N records.

`--head` `N`
: Save the first N records.

--tail N
: Save the last N records.

-b, --burst
: Save single record files in directory.

-u, --unpack
: Unpack records to files in directory.

-f, --force
: Replace existing file or directory.

-k, --keep
: Forward records to next command.

--help
: Show this message and exit.

Examples

Create an NFC Forum Text Record and save it to to a file in the /tmp directory, overwriting the file if it exists.

```
$ ndeftool text "Hello World" save --force /tmp/hello.ndef
Saving 1 record to /tmp/hello.ndef.
```

Same as above but the with three NDEF Text Records.

```
$ ndeftool text One text Two text Three save --force /tmp/text.ndef
Saving 3 records to /tmp/text.ndef.
```

Out of three records the second is saved using the `--skip` and `--count` options and the others are forwarded to print.

```
$ ndeftool txt aa txt bb txt cc save -f --skip 1 --count 1 /tmp/text.ndef print
Saving 1 record to /tmp/text.ndef.
NDEF Text Record ID '' Text 'aa' Language 'en' Encoding 'UTF-8'
NDEF Text Record ID '' Text 'cc' Language 'en' Encoding 'UTF-8'
```

Out of three records the second is saved using the `--head` and `--tail` options and the others are forwarded to print.

```
$ ndeftool txt aa txt bb txt cc save -f --head 2 --tail 1 /tmp/text.ndef print
Saving 1 record to /tmp/text.ndef.
NDEF Text Record ID '' Text 'aa' Language 'en' Encoding 'UTF-8'
NDEF Text Record ID '' Text 'cc' Language 'en' Encoding 'UTF-8'
```

Save each record to a separate file with auto-numbered file name plus `ndef` extension.

```
$ ndeftool txt aa txt bb txt cc save -f --burst /tmp/text/
Saving 1 record to /tmp/text/000.ndef.
Saving 1 record to /tmp/text/001.ndef.
Saving 1 record to /tmp/text/002.ndef.
```

Unpack record payloads to separate files using the record identifier as the file name.

```
$ ndeftool txt aa id 1.txt txt bb id 2.txt txt cc id 3.txt save -f --unpack /tmp/text/
Saving 1 record to /tmp/text/1.txt.
Saving 1 record to /tmp/text/2.txt.
Saving 1 record to /tmp/text/3.txt.
```

1.4.3 print

Print records as human readable.

Synopsis

```
ndeftool print [OPTIONS]
ndeftool p [OPTIONS]
```

Description

The **print** command outputs a formatted representation of all current NDEF Records. By default this is the one line str() representation for each record. The `--long` format produces multiple indented lines per record in an attempt to provide a more readable output. Printing consumes all records so that no more data is send to stdout or given to the next command. This can be changed with the `--keep` flag.

When given as the first command **print** attempts to decode an NDEF message from standard input and process the generated list of records.

Options

`-l, --long`
: Output in a long print format.

`-k, --keep`
: Keep records for next command.

`--help`
: Show this message and exit.

Examples

Print records in short format.

```
$ ndeftool text "Hello World" print
NDEF Text Record ID '' Text 'Hello World' Language 'en' Encoding 'UTF-8'
```

Print records in long format.

```
$ ndeftool text "Hello World" print --long
NFC Forum Text Record [record #1]
  content    Hello World
  language   en
  encoding   UTF-8
```

Print records in both short and long format.

```
$ ndeftool text "Hello World" print --keep print --long
NDEF Text Record ID '' Text 'Hello World' Language 'en' Encoding 'UTF-8'
NFC Forum Text Record [record #1]
  content    Hello World
  language   en
  encoding   UTF-8
```

1.4.4 identifier

Change the identifier of the last record.

Synopsis

```
ndeftool identifier [OPTIONS] NAME
ndeftool id [OPTIONS] NAME
```

Description

The **identifier** command either changes the current last record's name (NDEF Record ID) or, if the current message does not have any records, creates a record with unknown record type and the given record name.

Options

--help
Show this message and exit.

Examples

Create a record with `unknown` type and set the identifier.

```
$ ndeftool identifier 'record identifier' print
NDEF Record TYPE 'unknown' ID 'record identifier' PAYLOAD 0 byte
```

Create two text records with specific identifiers.

```
$ ndeftool text 'first' id 'r1' text 'second' id 'r2' print
NDEF Text Record ID 'r1' Text 'first' Language 'en' Encoding 'UTF-8'
NDEF Text Record ID 'r2' Text 'second' Language 'en' Encoding 'UTF-8'
```

1.4.5 typename

Change the type name of the last record.

Synopsis

```
ndeftool typename [OPTIONS] TYPE
ndeftool tn [OPTIONS] TYPE
```

Description

The **typename** command either changes the current last record's type (NDEF Record TNF and TYPE) or, if the current message does not have any records, creates a record with the given record type. The changed record is verified to successfully encode and decode unless disabled with `-x`.

Options

-x, --no-check
Do not check decoding after type name change.

--help
Show this message and exit.

Examples

Create a record with `text/plain` mime type and no payload.

```
$ ndeftool typename 'text/plain' print
NDEF Record TYPE 'text/plain' ID '' PAYLOAD 0 byte
```

Create a plain text record and add some payload.

```
$ ndeftool typename 'text/plain' payload 'Hello World' print
NDEF Record TYPE 'text/plain' ID '' PAYLOAD 11 byte '48656c6c6f20576f726c' ... 1 more
```

Create a record with a payload that does not match the record type.

```
$ ndeftool payload 'Hello World' typename -x 'urn:nfc:wkt:T' print -l
Record [record #1]
  type       urn:nfc:wkt:T
  name
  data       b'Hello World'
```

1.4.6 payload

Change the payload of the last record.

Synopsis

```
ndeftool payload [OPTIONS] DATA
ndeftool pl [OPTIONS] DATA
```

Description

The **payload** command either changes the current last record's data (NDEF Record PAYLOAD) or, if the current message does not have any records, creates a record with the given record data. The changed record is verified to successfully encode and decode unless disabled with `-x`.

The data string may contain hexadecimal bytes using `xNN` notation where each N is a nibble from [0-F].

Options

-x, --no-check
Do not check decoding after type name change.

--help
Show this message and exit.

Examples

Create a plain text payload record.

```
$ ndeftool payload 'Hello World' typename 'text/plain' print
NDEF Record TYPE 'text/plain' ID '' PAYLOAD 11 byte '48656c6c6f20576f726c' ... 1 more
```

Create an NFC Forum Text record with language code and content.

```
$ ndeftool payload '\x02enHello World' typename 'urn:nfc:wkt:T' print -l
NFC Forum Text Record [record #1]
  content    Hello World
  language   en
  encoding   UTF-8
```

Create a record with a payload that does not match the record type. The first command creates an NFC Forum Text Record with language code identifier and text content. The second command then replaces the payload with just the text and would make decoding fail.

```
$ ndeftool text 'Hello World' payload -x 'Hello World' print -l
Record [record #1]
  type       urn:nfc:wkt:T
  name
  data       b'Hello World'
```

1.4.7 text

Create an NFC Forum Text Record.

Synopsis

```
ndeftool text [OPTIONS] TEXT
ndeftool txt [OPTIONS] TEXT
```

Description

The **text** command creates an NFC Forum Text Record with the given input text. The text language defaults to English (language code `en`) and can be set with `--language` followed by the IANA language code. The text content is encoded as UTF-8 or UTF-16 depending on `--encoding`. The default encoding is UTF-8.

Options

-l, --language TEXT
: Set the IANA language code.

--encoding [UTF-8|UTF-16]
: Set the encoding (default UTF-8).

--help
: Show this message and exit.

Examples

Create an NFC Forum Text Record with the default language `en` and encoding `UTF-8`.

```
$ ndeftool text 'created with the nfcpy ndeftool' print
NDEF Text Record ID '' Text 'created with the nfcpy ndeftool' Language 'en' Encoding
↪'UTF-8'
```

Create one text record with English text and one record with German text.

```
$ ndeftool text --language en 'English' text --language de 'Deutsch' print
NDEF Text Record ID '' Text 'English' Language 'en' Encoding 'UTF-8'
NDEF Text Record ID '' Text 'Deutsch' Language 'de' Encoding 'UTF-8'
```

Create a text record with UTF-16 encoding.

```
$ ndeftool text --encoding UTF-16 'text encoded in UTF-16' | xxd -g 1
00000000: d1 01 31 54 82 65 6e ff fe 74 00 65 00 78 00 74  ..1T.en..t.e.x.t
00000010: 00 20 00 65 00 6e 00 63 00 6f 00 64 00 65 00 64  . .e.n.c.o.d.e.d
00000020: 00 20 00 69 00 6e 00 20 00 55 00 54 00 46 00 2d  . .i.n. .U.T.F.-
00000030: 00 31 00 36 00                                   .1.6.
```

1.4.8 uri

Create an NFC Forum URI Record.

Synopsis

```
ndeftool uri [OPTIONS] RESOURCE
```

Description

The **uri** command creates an NFC Forum URI Record with the given resource identifier. Note that this is actually an Internationalized Resource Identifier (IRI).

Options

--help
: Show this message and exit.

Examples

Create a URI record that links to `http://nfcpy.org`.

```
$ ndeftool uri 'http://nfcpy.org' print
NDEF Uri Record ID '' Resource 'http://nfcpy.org'
```

1.4.9 smartposter

Create an NFC Forum Smart Poster Record.

Synopsis

```
ndeftool smartposter [OPTIONS] RESOURCE
ndeftool smp [OPTIONS] RESOURCE
```

Description

The **smartposter** command creates an NFC Forum Smart Poster Record for the resource identifier. A smart poster record combines the uniform resource identifier with additional data such as titles and icons for representation and processing instructions for the reader application.

A smart poster record should have title text for the desired languages, added with repetitive `-t` options. An English title text may also be added with `-T`. The recommended action set with `-a` tells the reader application to either run the default action for the URI, save it for later or open for editing.

A smart poster may also provide a collection of icons for graphical representation. An icon file is added with the `-i` option that may be given more than once. The icon type is determined from the file content and must be an image or video mime type.

Options

-T TEXT
: Smartposter title for language code 'en'.

-t LANG TEXT
: Smartposter title for a given language code.

-a [exec|save|edit]
: Recommended action for handling the resource.

-i FILENAME
: Icon file for a graphical representation.

--help
: Show this message and exit.

Examples

An NFC Forum Smart Poster Record with just a link, nothing more useful than a URI Record.

```
$ ndeftool smartposter http://nfcpy.org print
NDEF Smartposter Record ID '' Resource 'http://nfcpy.org'
```

Same as above but with an English title.

```
$ ndeftool smartposter -T 'nfcpy project' http://nfcpy.org print
NDEF Smartposter Record ID '' Resource 'http://nfcpy.org' Title 'nfcpy project'
```

Titles for other languages must be given with a language code.

```
$ ndeftool smartposter -t de 'Google Deutschland' https://www.google.de print
NDEF Smartposter Record ID '' Resource 'https://www.google.de' Title 'Google␣
↪Deutschland'
```

An emergency call number should be called immediately.

```
$ ndeftool smartposter -T 'EMERGENCY CALL 911' -a exec 'tel:911' print -l
NFC Forum Smart Poster Record [record #1]
  resource   tel:911
  action     exec
  title_en   EMERGENCY CALL 911
```

Add an icon file to a smart poster.

```
$ ndeftool smp -i images/ndeftool.png https://github.com/nfcpy/ndeftool print -l
NFC Forum Smart Poster Record [record #1]
  resource   https://github.com/nfcpy/ndeftool
  image/png  19941 byte
```

1.5 Examples

Any NDEF Record can be constructed with the *payload*, *typename* and *identifier* commands.

```
$ ndeftool payload '\02ensample text' typename 'urn:nfc:wkt:T' id 'r1' print
NDEF Text Record ID 'r1' Text 'sample text' Language 'en' Encoding 'UTF-8'
```

The same record can be created with the *text* command. Here the output goes to stdout and is then printed with a separate ndeftool process call.

```
$ ndeftool text 'sample text' id 'r1' | ndeftool print
Reading data from standard input
NDEF Text Record ID 'r1' Text 'sample text' Language 'en' Encoding 'UTF-8'
```

The *save* command writes the records to disk or <stdout> for path name −. The following example creates an NDEF message with three NFC Forum Text Records, the first and last record with the message begin and message end flags set.

```
$ ndeftool text ONE text TWO text THREE save - | xxd -g 1
Saving 3 records to <stdout>.
00000000: 91 01 06 54 02 65 6e 4f 4e 45 11 01 06 54 02 65  ...T.enONE...T.e
00000010: 6e 54 57 4f 51 01 08 54 02 65 6e 54 48 52 45 45  nTWOQ..T.enTHREE
```

The *save* command can be used to write intermediate results, here immediately after a text record has been created. Note that by writing to <stdout> the result is a sequence of three individual NDEF messages of one record each. This would not be a proper NDEF message file.

```
$ ndeftool text ONE save - text TWO save - text THREE save - | xxd -g 1
Saving 1 record to <stdout>.
Saving 1 record to <stdout>.
Saving 1 record to <stdout>.
00000000: d1 01 06 54 02 65 6e 4f 4e 45 d1 01 06 54 02 65  ...T.enONE...T.e
00000010: 6e 54 57 4f d1 01 08 54 02 65 6e 54 48 52 45 45  nTWO...T.enTHREE
```

The *load* command reads records from disk or <stdin> for path name −.

```
$ ndeftool text ONE text TWO text THREE | ndeftool load - print
loaded 3 record(s) from <stdin>
NDEF Text Record ID '' Text 'ONE' Language 'en' Encoding 'UTF-8'
NDEF Text Record ID '' Text 'TWO' Language 'en' Encoding 'UTF-8'
NDEF Text Record ID '' Text 'THREE' Language 'en' Encoding 'UTF-8'
```

An empty NDEF Record can be created with an empty type name string. The first octet `11010000b` sets the Message Begin (MB) and Message End (ME) flags in the two most signifant bits. The Type Name Format (TNF) value 0 in the least significant three bits indicates that there is no type or payload associated with this record and thus the `TYPE LENGTH` and `PAYLOAD LENGTH` fields must be zero.

```
$ ndeftool typename '' | xxd -g 1
00000000: d0 00 00                                         ...
```

The default decoding of an NDEF message requires correct data format. Data with minor format errors can be decoded with the `--relax` option. The following example creates two empty records with invalid MB and ME flags that do only decode with `--relax`.

```
$ python3 -c "import sys; sys.stdout.buffer.write(b'\x10\0\0\x10\0\0')" | ndeftool --
↪relax print
Reading data from standard input
NDEF Record TYPE '' ID '' PAYLOAD 0 byte
NDEF Record TYPE '' ID '' PAYLOAD 0 byte
```

NDEF message data with uncorrectable errors can be skipped with the `--ignore` option. The payload length 1 in the second record is an invalid value for an empty record.

```
$ python3 -c "import sys; sys.stdout.buffer.write(b'\x10\0\0\x10\1\0')" | ndeftool --
↪ignore print
Reading data from standard input
NDEF Record TYPE '' ID '' PAYLOAD 0 byte
```

CHAPTER 2

Contributing

Thank you for considering contributing to **ndeftool**. There are many ways to help and any help is welcome.

2.1 Reporting issues

- Under which versions of Python does this happen? This is especially important if your issue is encoding related.
- Under which versions of **ndeftool** does this happen? Check if this issue is fixed in the repository.

2.2 Submitting patches

- Include tests if your patch is supposed to solve a bug, and explain clearly under which circumstances the bug happens. Make sure the test fails without your patch.
- Include or update tests and documentation if your patch is supposed to add a new feature. Note that documentation is in two places, the code itself for rendering help pages and in the docs folder for the online documentation.
- Follow PEP 8 and PEP 257.

2.3 Development tips

- Fork the repository and clone it locally:

```
git clone git@github.com:your-username/ndeftool.git
cd ndeftool
```

- Create virtual environments for Python 2 an Python 3, setup the ndeftool package in develop mode, and install required development packages:

```
virtualenv python-2
python3 -m venv python-3
source python-2/bin/activate
python setup.py develop
pip install -r requirements-dev.txt
source python-3/bin/activate
python setup.py develop
pip install -r requirements-dev.txt
```

- Verify that all tests pass and the documentation is build:

```
tox
```

- Preferably develop in the Python 3 virtual environment. Running `tox` ensures tests are run with both the Python 2 and Python 3 interpreter but it takes some time to complete. Alternatively switch back and forth between versions and just run the tests:

```
source python-2/bin/activate
py.test
source python-3/bin/activate
py.test
```

- Test coverage should be close to 100 percent. A great help is the HTML output produced by coverage.py:

```
py.test --cov ndeftool --cov-report html
firefox htmlcov/index.html
```

- The documentation can be created and viewed loacally:

```
(cd docs && make html)
firefox docs/_build/html/index.html
```

CHAPTER 3

License

The **ndeftool** is licensed under the Internet Systems Consortium ISC license. This is a permissive free software license functionally equivalent to the simplified BSD and the MIT license.

3.1 License text

Index

Symbols

C