MT940 Documentation

Release 4.30.0

Rick van Hattem (wolph)

Apr 13, 2023

Contents

Contents:

CHAPTER 1

Installation

At the command line:

```
$ easy_install mt940
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv mt940
$ pip install mt940
```

CHAPTER 2

Usage

To use MT940 in a project:

```
import mt940
```

CHAPTER 3

mt940

3.1 mt940 package

class mt940.**JSONEncoder**(*, *skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *sort_keys=False*, *indent=None*, *separators=None*, *default=None*)

Bases: json.encoder.JSONEncoder

default(*value*)

Implement this method in a subclass such that it returns a serializable object for o, or calls the base implementation (to raise a TypeError).

For example, to support arbitrary iterators, you could implement default like this:

```
def default(self, o):
    try:
        iterable = iter(o)
    except TypeError:
        pass
    else:
        return list(iterable)
    # Let the base class default method raise the TypeError
    return JSONEncoder.default(self, o)
```

mt940.**parse**(*src*, *encoding=None*, *processors=None*, *tags=None*)

Parses mt940 data and returns transactions object

Parameters **src** – file handler to read, filename to read or raw data as string

Returns Collection of transactions

Return type *Transactions*

3.1.1 Submodules

mt940.json module

class mt940.json.**JSONEncoder**(*, *skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *sort_keys=False*, *indent=None*, *separators=None*, *default=None*)

Bases: json.encoder.JSONEncoder

default(*value*)

Implement this method in a subclass such that it returns a serializable object for o, or calls the base implementation (to raise a TypeError).

For example, to support arbitrary iterators, you could implement default like this:

```
def default(self, o):
    try:
        iterable = iter(o)
    except TypeError:
        pass
    else:
        return list(iterable)
    # Let the base class default method raise the TypeError
    return JSONEncoder.default(self, o)
```

mt940.models module

class mt940.models.**Amount**(*amount*, *status*, *currency=None*, ***kwargs*)

Bases: *mt940.models.Model*

Amount object containing currency and amount

Parameters

- **amount** (*str*) – Amount using either a , or a . as decimal separator
- **status** (*str*) – Either C or D for credit or debit respectively
- **currency** (*str*) – A 3 letter currency (e.g. EUR)

```
>>> Amount('123.45', 'C', 'EUR')
<123.45 EUR>
>>> Amount('123.45', 'D', 'EUR')
<-123.45 EUR>
```

class mt940.models.**Balance**(*status=None*, *amount=None*, *date=None*, ***kwargs*)

Bases: *mt940.models.Model*

Parse balance statement

Parameters

- **status** (*str*) – Either C or D for credit or debit respectively
- **amount** (Amount) – Object containing the amount and currency
- **date** (*date*) – The balance date

```
>>> balance = Balance('C', '0.00', Date(2010, 7, 22))
>>> balance.status
'C'
>>> balance.amount.amount
```

(continues on next page)

(continued from previous page)

```
Decimal('0.00')
>>> isinstance(balance.date, Date)
True
>>> balance.date.year, balance.date.month, balance.date.day
(2010, 7, 22)
```

```
>>> Balance()
<None @ None>
```

class mt940.models.**Date**

Bases: datetime.date, *mt940.models.Model*

Just a regular date object which supports dates given as strings

```
>>> Date(year='2000', month='1', day='2')
Date(2000, 1, 2)
```

```
>>> Date(year='123', month='1', day='2')
Date(2123, 1, 2)
```

Parameters

- **year** (*str*) – Year (0-100), will automatically add 2000 when needed
- **month** (*str*) – Month
- **day** (*str*) – Day

class mt940.models.**DateTime**

Bases: datetime.datetime, *mt940.models.Model*

Just a regular datetime object which supports dates given as strings

```
>>> DateTime(year='2000', month='1', day='2', hour='3', minute='4',
...          second='5', microsecond='6')
DateTime(2000, 1, 2, 3, 4, 5, 6)
```

```
>>> DateTime(year='123', month='1', day='2', hour='3', minute='4',
...          second='5', microsecond='6')
DateTime(2123, 1, 2, 3, 4, 5, 6)
```

```
>>> DateTime(2000, 1, 2, 3, 4, 5, 6)
DateTime(2000, 1, 2, 3, 4, 5, 6)
```

```
>>> DateTime(year='123', month='1', day='2', hour='3', minute='4',
...          second='5', microsecond='6', tzinfo=FixedOffset('60'))
DateTime(2123, 1, 2, 3, 4, 5, 6, tzinfo=<mt940.models.FixedOffset ...>)
```

Parameters

- **year** (*str*) – Year (0-100), will automatically add 2000 when needed
- **month** (*str*) – Month
- **day** (*str*) – Day
- **hour** (*str*) – Hour

- **minute** (*str*) – Minute
- **second** (*str*) – Second
- **microsecond** (*str*) – Microsecond
- **tzinfo** (*tzinfo*) – Timezone information. Overwrites *offset*
- **offset** (*str*) – Timezone offset in minutes, generates a tzinfo object with the given offset if no tzinfo is available.

class mt940.models.**FixedOffset**(*offset=0*, *name=None*)
Bases: datetime.tzinfo

Fixed time offset based on the Python docs Source: https://docs.python.org/2/library/datetime.html#tzinfo-objects

```
>>> offset = FixedOffset(60)
>>> offset.utcoffset(None).total_seconds()
3600.0
>>> offset.dst(None)
datetime.timedelta(0)
>>> offset.tzname(None)
'60'
```

dst(*dt*)
datetime -> DST offset as timedelta positive east of UTC.

tzname(*dt*)
datetime -> string name of time zone.

utcoffset(*dt*)
datetime -> timedelta showing offset from UTC, negative values indicating West of UTC

class mt940.models.**Model**
Bases: object

class mt940.models.**SumAmount**(**args*, ***kwargs*)
Bases: *mt940.models.Amount*

class mt940.models.**Transaction**(*transactions*, *data=None*)
Bases: *mt940.models.Model*

update(*data*)

class mt940.models.**Transactions**(*processors=None*, *tags=None*)
Bases: collections.abc.Sequence

Collection of *Transaction* objects with global properties such as begin and end balance

DEFAULT_PROCESSORS = {'post_account_identification': [], 'post_available_balance': [
Using the processors you can pre-process data before creating objects and modify them after creating the objects

currency

static defaultTags()

classmethod normalize_tag_id(*tag_id*)

parse(*data*)
Parses mt940 data, expects a string with data

Parameters **data** (*str*) – The MT940 data

Returns: `list` of *`Transaction`*

`sanitize_tag_id_matches`(*matches*)

`classmethod strip`(*lines*)

mt940.parser module

Format

Sources:

- Swift for corporates
- Rabobank MT940

```
[] = optional
! = fixed length
a = Text
x = Alphanumeric, seems more like text actually. Can include special
    characters (slashes) and whitespace as well as letters and numbers
d = Numeric separated by decimal (usually comma)
c = Code list value
n = Numeric
```

mt940.parser.**`parse`**(*src*, *encoding=None*, *processors=None*, *tags=None*)

Parses mt940 data and returns transactions object

Parameters **`src`** – file handler to read, filename to read or raw data as string

Returns Collection of transactions

Return type *Transactions*

mt940.processors module

mt940.processors.**`add_currency_pre_processor`**(*currency*, *overwrite=True*)

mt940.processors.**`date_cleanup_post_processor`**(*transactions*, *tag*, *tag_dict*, *result*)

mt940.processors.**`date_fixup_pre_processor`**(*transactions*, *tag*, *tag_dict*, **args*)

Replace illegal February 29, 30 dates with the last day of February.

German banks use a variant of the 30/360 interest rate calculation, where each month has always 30 days even February. Python's datetime module won't accept such dates.

mt940.processors.**`mBank_set_iph_id`**(*transactions*, *tag*, *tag_dict*, **args*)

mBank Collect uses ID IPH to distinguish between virtual accounts, adding iph_id may be helpful in further processing

mt940.processors.**`mBank_set_tnr`**(*transactions*, *tag*, *tag_dict*, **args*)

mBank Collect states TNR in transaction details as unique id for transactions, that may be used to identify the same transactions in different statement files eg. partial mt942 and full mt940 Information about tnr uniqueness has been obtained from mBank support, it lacks in mt940 mBank specification.

mt940.processors.**`mBank_set_transaction_code`**(*transactions*, *tag*, *tag_dict*, **args*)

mBank Collect uses transaction code 911 to distinguish icoming mass payments transactions, adding transaction_code may be helpful in further processing

mt940.processors.**transaction_details_post_processor**(*transactions*, *tag*, *tag_dict*, *result*, *space=False*)

Parse the extra details in some transaction formats such as the 60-65 keys.

Parameters

- **transactions** (mt940.models.Transactions) – list of transactions
- **tag** (mt940.tags.Tag) – tag
- **tag_dict** (*dict*) – dict with the raw tag details
- **result** (*dict*) – the resulting tag dict
- **space** (*bool*) – include spaces between lines in the mt940 details

mt940.processors.**transaction_details_post_processor_with_space**(*transactions*, *tag*, *tag_dict*, *result*, ***, *space=True*)

Parse the extra details in some transaction formats such as the 60-65 keys.

Parameters

- **transactions** (mt940.models.Transactions) – list of transactions
- **tag** (mt940.tags.Tag) – tag
- **tag_dict** (*dict*) – dict with the raw tag details
- **result** (*dict*) – the resulting tag dict
- **space** (*bool*) – include spaces between lines in the mt940 details

mt940.processors.**transactions_to_transaction**(**keys*)

Copy the global transactions details to the transaction.

Parameters ***keys** (*str*) – the keys to copy to the transaction

mt940.tags module

The MT940 format is a standard for bank account statements. It is used by many banks in Europe and is based on the SWIFT MT940 format.

The MT940 tags are:

Tag	Description
:13:	Date/Time indication at which the report was created
:20:	Transaction Reference Number
:21:	Related Reference Number
:25:	Account Identification
:28:	Statement Number
:34:	The floor limit for debit and credit
:60F:	Opening Balance
:60M:	Intermediate Balance
:60E:	Closing Balance
:61:	Statement Line
:62:	Closing Balance
:62M:	Intermediate Closing Balance
:62F:	Final Closing Balance
:64:	Available Balance
:65:	Forward Available Balance
:86:	Transaction Information
:90:	Total number and amount of debit entries
:NS:	Bank specific Non-swift extensions containing extra information

Format

Sources:

- Swift for corporates
- Rabobank MT940

The pattern for the tags use the following syntax:

```
[] = optional
! = fixed length
a = Text
x = Alphanumeric, seems more like text actually. Can include special
    characters (slashes) and whitespace as well as letters and numbers
d = Numeric separated by decimal (usually comma)
c = Code list value
n = Numeric
```

class mt940.tags.**AccountIdentification**

Bases: *mt940.tags.Tag*

Account identification

Pattern: 35x

id = 25

logger = <Logger mt940.tags.AccountIdentification (WARNING)>

name = 'AccountIdentification'

pattern = '(?P<account_identification>.{0,35})'

slug = 'account_identification'

class mt940.tags.**AvailableBalance**

Bases: *mt940.tags.BalanceBase*

id = 64

logger = <Logger mt940.tags.AvailableBalance (WARNING)>

name = 'AvailableBalance'

slug = 'available_balance'

class mt940.tags.**BalanceBase**
Bases: *mt940.tags.Tag*

Balance base

Pattern: 1!a6!n3!a15d

pattern = '^\n (?P<status>[DC]) # 1!a Debit/Credit\n (?P<year>\\d{2}) # 6!n Value Date

class mt940.tags.**ClosingBalance**
Bases: *mt940.tags.BalanceBase*

id = 62

logger = <Logger mt940.tags.ClosingBalance (WARNING)>

name = 'ClosingBalance'

slug = 'closing_balance'

class mt940.tags.**DateTimeIndication**
Bases: *mt940.tags.Tag*

Date/Time indication at which the report was created

Pattern: 6!n4!n1! x4!n

id = 13

logger = <Logger mt940.tags.DateTimeIndication (WARNING)>

name = 'DateTimeIndication'

pattern = '^\n (?P<year>\\d{2})\n (?P<month>\\d{2})\n (?P<day>\\d{2})\n (?P<hour>\\d{2

slug = 'date_time_indication'

class mt940.tags.**FinalClosingBalance**
Bases: *mt940.tags.ClosingBalance*

id = '62F'

logger = <Logger mt940.tags.FinalClosingBalance (WARNING)>

name = 'FinalClosingBalance'

slug = 'final_closing_balance'

class mt940.tags.**FinalOpeningBalance**
Bases: *mt940.tags.BalanceBase*

id = '60F'

logger = <Logger mt940.tags.FinalOpeningBalance (WARNING)>

name = 'FinalOpeningBalance'

slug = 'final_opening_balance'

class mt940.tags.**FloorLimitIndicator**
Bases: *mt940.tags.Tag*

Floor limit indicator indicates the minimum value reported for debit and credit amounts

Pattern: :34F:GHSC0,00

id = 34

logger = <Logger mt940.tags.FloorLimitIndicator (WARNING)>

name = 'FloorLimitIndicator'

pattern = '^\n (?P<currency>[A-Z]{3}) # 3!a Currency\n (?P<status>[DC]?) # 2a Debit/C

slug = 'floor_limit_indicator'

class mt940.tags.**ForwardAvailableBalance**
Bases: *mt940.tags.BalanceBase*

id = 65

logger = <Logger mt940.tags.ForwardAvailableBalance (WARNING)>

name = 'ForwardAvailableBalance'

slug = 'forward_available_balance'

class mt940.tags.**IntermediateClosingBalance**
Bases: *mt940.tags.ClosingBalance*

id = '62M'

logger = <Logger mt940.tags.IntermediateClosingBalance (WARNING)>

name = 'IntermediateClosingBalance'

slug = 'intermediate_closing_balance'

class mt940.tags.**IntermediateOpeningBalance**
Bases: *mt940.tags.BalanceBase*

id = '60M'

logger = <Logger mt940.tags.IntermediateOpeningBalance (WARNING)>

name = 'IntermediateOpeningBalance'

slug = 'intermediate_opening_balance'

class mt940.tags.**NonSwift**
Bases: *mt940.tags.Tag*

Non-swift extension for MT940 containing extra information. The actual definition is not consistent between banks so the current implementation is a tad limited. Feel free to extend the implementation and create a pull request with a better version :)

It seems this could be anything so we'll have to be flexible about it.

Pattern: *2!n35x | *x*

id = 'NS'

logger = <Logger mt940.tags.NonSwift (WARNING)>

name = 'NonSwift'

pattern = '\n (?P<non_swift>\n (\n (\\d{2}.{0,})\n (\\n\\d{2}.{0,})*\n)|(\n [^\\n]*\n

class scope(*transactions*, *data=None*)
Bases: *mt940.models.Transaction*, *mt940.models.Transactions*

slug = 'non_swift'

sub_pattern = '\n (?P<ns_id>\\d{2})(?P<ns_data>.{0,})\n '

sub_pattern_m = re.compile('\n (?P<ns_id>\\d{2})(?P<ns_data>.{0,})\n ', re.IGNORECASE|

class mt940.tags.**OpeningBalance**
Bases: *mt940.tags.BalanceBase*

id = 60

logger = <Logger mt940.tags.OpeningBalance (WARNING)>

name = 'OpeningBalance'

slug = 'opening_balance'

class mt940.tags.**RelatedReference**
Bases: *mt940.tags.Tag*

Related reference

Pattern: 16x

id = 21

logger = <Logger mt940.tags.RelatedReference (WARNING)>

name = 'RelatedReference'

pattern = '(?P<related_reference>.{0,16})'

slug = 'related_reference'

class mt940.tags.**Statement**
Bases: *mt940.tags.Tag*

The MT940 Tag 61 provides information about a single transaction that has taken place on the account. Each transaction is identified by a unique transaction reference number (Tag 20) and is described in the Statement Line (Tag 61).

Pattern: 6!n[4!n]2a[1!a]15d1!a3!c23x[//16x]

The fields are:

- *value_date*: transaction date (YYMMDD)
- *entry_date*: Optional 4-digit month value and 2-digit day value of the entry date (MMDD)
- *funds_code*: Optional 1-character code indicating the funds type (the third character of the currency code if needed)
- *amount*: 15-digit value of the transaction amount, including commas for decimal separation
- *transaction_type*: Optional 4-character transaction type identification code starting with a letter followed by alphanumeric characters and spaces
- *customer_reference*: Optional 16-character customer reference, excluding any bank reference
- *bank_reference*: Optional 23-character bank reference starting with "//"
- *supplementary_details*: Optional 34-character supplementary details about the transaction.

The Tag 61 can occur multiple times within an MT940 file, with each occurrence representing a different transaction.

id = 61

logger = <Logger mt940.tags.Statement (WARNING)>

name = 'Statement'

pattern = "^\n (?P<year>\\d{2}) # 6!n Value Date (YYMMDD)\n (?P<month>\\d{2})\n (?P<da

scope
alias of *mt940.models.Transaction*

slug = 'statement'

class mt940.tags.**StatementASNB**
Bases: *mt940.tags.Statement*

From: https://www.sepaforcorporates.com/swift-for-corporates

Pattern: 6!n[4!n]2a[1!a]15d1!a3!c16x[//16x] [34x]

But ASN bank puts the IBAN in the customer reference, which is acording to Wikipedia at most 34 characters.

So this is the new pattern:

Pattern: 6!n[4!n]2a[1!a]15d1!a3!c34x[//16x] [34x]

pattern = '^\n (?P<year>\\d{2}) # 6!n Value Date (YYMMDD)\n (?P<month>\\d{2})\n (?P<da

class mt940.tags.**StatementNumber**
Bases: *mt940.tags.Tag*

Statement number / sequence number

Pattern: 5n[/5n]

id = 28

logger = <Logger mt940.tags.StatementNumber (WARNING)>

name = 'StatementNumber'

pattern = '\n (?P<statement_number>\\d{1,5}) # 5n\n (?:/?(?P<sequence_number>\\d{1,5})

slug = 'statement_number'

class mt940.tags.**SumCreditEntries**
Bases: *mt940.tags.SumEntries*

id = '90C'

logger = <Logger mt940.tags.SumCreditEntries (WARNING)>

name = 'SumCreditEntries'

slug = 'sum_credit_entries'

status = 'C'

class mt940.tags.**SumDebitEntries**
Bases: *mt940.tags.SumEntries*

id = '90D'

logger = <Logger mt940.tags.SumDebitEntries (WARNING)>

name = 'SumDebitEntries'

slug = 'sum_debit_entries'

status = 'D'

class mt940.tags.**SumEntries**
Bases: *mt940.tags.Tag*

Number and Sum of debit Entries

id = 90

logger = <Logger mt940.tags.SumEntries (WARNING)>

name = 'SumEntries'

pattern = '^\n (?P<number>\\d*)\n (?P<currency>.{3}) # 3!a Currency\n (?P<amount>[\\d,

slug = 'sum_entries'

class mt940.tags.**Tag**
Bases: object

RE_FLAGS = 98

id = 0

parse(*transactions*, *value*)

scope
alias of *mt940.models.Transactions*

class mt940.tags.**Tags**
Bases: enum.Enum

An enumeration.

ACCOUNT_IDENTIFICATION = <mt940.tags.AccountIdentification object>

AVAILABLE_BALANCE = <mt940.tags.AvailableBalance object>

CLOSING_BALANCE = <mt940.tags.ClosingBalance object>

DATE_TIME_INDICATION = <mt940.tags.DateTimeIndication object>

FINAL_CLOSING_BALANCE = <mt940.tags.FinalClosingBalance object>

FINAL_OPENING_BALANCE = <mt940.tags.FinalOpeningBalance object>

FLOOR_LIMIT_INDICATOR = <mt940.tags.FloorLimitIndicator object>

FORWARD_AVAILABLE_BALANCE = <mt940.tags.ForwardAvailableBalance object>

INTERMEDIATE_CLOSING_BALANCE = <mt940.tags.IntermediateClosingBalance object>

INTERMEDIATE_OPENING_BALANCE = <mt940.tags.IntermediateOpeningBalance object>

NON_SWIFT = <mt940.tags.NonSwift object>

OPENING_BALANCE = <mt940.tags.OpeningBalance object>

RELATED_REFERENCE = <mt940.tags.RelatedReference object>

STATEMENT = <mt940.tags.Statement object>

STATEMENT_NUMBER = <mt940.tags.StatementNumber object>

SUM_CREDIT_ENTRIES = <mt940.tags.SumCreditEntries object>

SUM_DEBIT_ENTRIES = <mt940.tags.SumDebitEntries object>

SUM_ENTRIES = <mt940.tags.SumEntries object>

TRANSACTION_DETAILS = <mt940.tags.TransactionDetails object>

TRANSACTION_REFERENCE_NUMBER = <mt940.tags.TransactionReferenceNumber object>

class mt940.tags.**TransactionDetails**

Bases: *mt940.tags.Tag*

Transaction details

Pattern: 6x65x

id = 86

logger = <Logger mt940.tags.TransactionDetails (WARNING)>

name = 'TransactionDetails'

pattern = '\n (?P<transaction_details>(([\\s\\S]{0,65}\\r?\\n?){0,8}[\\s\\S]{0,65}))\n

scope

alias of *mt940.models.Transaction*

slug = 'transaction_details'

class mt940.tags.**TransactionReferenceNumber**

Bases: *mt940.tags.Tag*

Transaction reference number

Pattern: 16x

id = 20

logger = <Logger mt940.tags.TransactionReferenceNumber (WARNING)>

name = 'TransactionReferenceNumber'

pattern = '(?P<transaction_reference>.{0,16})'

slug = 'transaction_reference_number'

mt940.utils module

class mt940.utils.**Strip**

Bases: enum.IntEnum

An enumeration.

BOTH = 3

LEFT = 1

NONE = 0

RIGHT = 2

mt940.utils.**coalesce**(**args*)

Return the first non-None argument

```
>>> coalesce()
```

```
>>> coalesce(0, 1)
0
>>> coalesce(None, 0)
0
```

mt940.utils.**join_lines**(*string*, *strip=<Strip.BOTH: 3>*)
Join strings together and strip whitespace in between if needed

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/WoLpH/mt940/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

4.1.4 Write Documentation

MT940 could always use more documentation, whether as part of the official MT940 docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/WoLpH/mt940/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *mt940* for local development.

1. Fork the *mt940* repo on GitHub.
2. Clone your fork locally:

```
$ git clone --branch develop git@github.com:your_name_here/mt940.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv mt940
$ cd mt940/
$ pip install -e .
```

4. Create a branch for local development with git-flow-avh:

```
$ git-flow feature start name-of-your-bugfix-or-feature
```

   Or without git-flow:

   $ git checkout -b feature/name-of-your-bugfix-or-feature

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 mt940 mt940_tests
$ py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv using the requirements file.

   $ pip install -r mt940_tests/requirements.txt

6. Commit your changes and push your branch to GitHub with git-flow-avh:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git-flow feature publish
```

Or without git-flow:

$ git add . $ git commit -m “Your detailed description of your changes.” $ git push -u origin feature/name-of-your-bugfix-or-feature

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4+, and for PyPy. Check https://travis-ci.org/WoLpH/mt940/pull_requests and make sure that the tests pass for all supported Python versions. To test locally you can use *tox* which will run on all installed Python versions.

4.4 Tips

To run a subset of tests:

```
$ py.test mt940_tests/some_test.py
```

CHAPTER 5

Credits

5.1 Development Lead

- Rick van Hattem (Wolph) <wolph@wol.ph>

5.2 Contributors

- Ben Konrath (benkonrath)

Why not join the club?

A more up to date list can be found here: https://github.com/WoLpH/mt940/graphs/contributors

CHAPTER 6

History

A complete list of changes can be read through the commit log: https://github.com/WoLpH/mt940/commits/develop

The list of releases (with changelog) can be found here: https://github.com/WoLpH/mt940/releases

- **Documentation**
  - http://mt940.readthedocs.org/en/latest/
- **Source**
  - https://github.com/WoLpH/mt940
- **Bug reports**
  - https://github.com/WoLpH/mt940/issues
- **Package homepage**
  - https://pypi.python.org/pypi/mt-940
- **My blog**
  - http://wol.ph/

CHAPTER 7

Install

To install the latest release:

```
pip install mt-940
```

Or if *pip* is not available:

```
easy_install mt-940
```

To install the latest development release:

```
git clone --branch develop https://github.com/WoLpH/mt940.git mt940
cd ./mt940
virtualenv .env
source .env/bin/activate
pip install -e .
```

To run the tests you can use the *py.test* command or just run *tox* to test everything in all supported python versions.

CHAPTER 8

Usage

Basic parsing:

```
import mt940
import pprint

transactions = mt940.parse('mt940_tests/jejik/abnamro.sta')

print('Transactions:')
print(transactions)
pprint.pprint(transactions.data)

print()
for transaction in transactions:
    print('Transaction: ', transaction)
    pprint.pprint(transaction.data)
```

Set opening / closing balance information on each transaction:

```
import mt940
import pprint

mt940.tags.BalanceBase.scope = mt940.models.Transaction

# The currency has to be set manually when setting the BalanceBase scope to␣
↪Transaction.
transactions = mt940.models.Transactions(processors=dict(
    pre_statement=[
        mt940.processors.add_currency_pre_processor('EUR'),
    ],
))

with open('mt940_tests/jejik/abnamro.sta') as f:
    data = f.read()
```

(continues on next page)

(continued from previous page)

```
transactions.parse(data)

for transaction in transactions:
    print('Transaction: ', transaction)
    pprint.pprint(transaction.data)
```

Simple json encoding:

```
import json
import mt940


transactions = mt940.parse('mt940_tests/jejik/abnamro.sta')

print(json.dumps(transactions, indent=4, cls=mt940.JSONEncoder))
```

Parsing statements from the Dutch bank ASN where tag 61 does not follow the Swift specifications:

```
def ASNB_mt940_data():
    with open('mt940_tests/ASNB/0708271685_09022020_164516.940.txt') as fh:
        return fh.read()

def test_ASNB_tags(ASNB_mt940_data):
    tag_parser = mt940.tags.StatementASNB()
    trs = mt940.models.Transactions(tags={
        tag_parser.id: tag_parser
    })

trs.parse(ASNB_mt940_data)
trs_data = pprint.pformat(trs.data, sort_dicts=False)
print(trs_data)
```

CHAPTER 9

Contributing

Help is greatly appreciated, just please remember to clone the **development** branch and to run *tox* before creating pull requests.

Travis tests for *flake8* support and test coverage so it's always good to check those before creating a pull request.

Development branch: https://github.com/WoLpH/mt940/tree/develop

To run the tests:

```
pip install -r mt940_tests/requirements.txt
py.test
```

Or to run the tests on all available Python versions:

```
pip install tox
tox
```

CHAPTER 10

Info

Python support	Python 2.7, >= 3.3
Blog	http://wol.ph/
Source	https://github.com/WoLpH/mt940
Documentation	http://mt940.rtfd.org
Changelog	http://mt940.readthedocs.org/en/latest/history.html
API	http://mt940.readthedocs.org/en/latest/modules.html
Issues/roadmap	https://github.com/WoLpH/mt940/issues
Travis	http://travis-ci.org/WoLpH/mt940
Test coverage	https://coveralls.io/r/WoLpH/mt940
Pypi	https://pypi.python.org/pypi/mt-940
Ohloh	https://www.ohloh.net/p/mt-940
License	BSD.
git repo	`$ git clone https://github.com/WoLpH/mt940.git`

install dev	`$ git clone https://github.com/WoLpH/mt940.git mt940`
	`$ cd ./mt940`
	`$ virtualenv .env`
	`$ source .env/bin/activate`
	`$ pip install -e .`

tests	`$ py.test`

CHAPTER 11

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index

J

L

M

N

O

P

R

S

T

U