iso8583

Release 3.0.0

Jun 05, 2023

Table of Contents:

1 At a Glance **3**
- 1.1 Intro . . . 3
- 1.2 How-To . . . 5
- 1.3 Specifications . . . 10
- 1.4 Functions . . . 22
- 1.5 Change Log . . . 25

Python Module Index **27**

Index **29**

`iso8583` is a Python package that serializes and deserializes ISO8583 data between a `bytes` or `bytearray` instance containing ISO8583 data and a Python `dict`.

CHAPTER 1

At a Glance

`iso8583` package supports custom specifications. See *iso8583.specs* module.

Use *iso8583.decode()* to decode raw iso8583 message.

```
>>> import iso8583
>>> from iso8583.specs import default_ascii as spec
>>> encoded_raw = b'0200400000000000000001012345678900'
>>> decoded, encoded = iso8583.decode(encoded_raw, spec)
```

Use *iso8583.encode()* to encode updated ISO8583 message. It returns a raw ISO8583 message and a dictionary with encoded data.

```
>>> import iso8583
>>> from iso8583.specs import default_ascii as spec
>>> decoded = {"t": "0200", "2": "1234567890", "39": "00"}
>>> encoded_raw, encoded = iso8583.encode(decoded, spec)
```

To install:

```
pip install pyiso8583
```

1.1 Intro

1.1.1 iso8583 - a Python package for parsing ISO8583 data

`iso8583` package serializes and deserializes ISO8583 data between raw `bytes` ISO8583 data and a regular Python `dict`.

`iso8583` package supports custom specifications that can define:

- Field length and data encoding, such as BCD, ASCII, EBCDIC, etc.

- Field length count measured in bytes or nibbles.
- Field type, such as fixed, LLVAR, LLLVAR, etc.
- Maximum length
- Optional field description

Multiple specifications can co-exist to support ISO8583 messages for POS, ATM, file actions, and so on. Simply define a new specification dictionary. `iso8583` package includes a starter specification in `iso8583.specs` module that can be used as a base to create own custom/proprietary specifications.

Additional information is available on Read The Docs.

Installation

`iso8583` is published on PyPI as `pyiso8583` and can be installed from there:

Encoding & Decoding

Use iso8583.decode to decode raw ISO8583 message. It returns two dictionaries: one with decoded data and one with encoded data.

```
>>> import pprint
>>> import iso8583
>>> from iso8583.specs import default_ascii as spec
>>> encoded_raw = b'0200400000000000000001012345678900'
>>> decoded, encoded = iso8583.decode(encoded_raw, spec)
>>> pprint.pp(decoded)
{'t': '0200', 'p': '4000000000000000', '2': '1234567890'}
>>> pprint.pp(encoded)
{'t': {'len': b'', 'data': b'0200'},
 'p': {'len': b'', 'data': b'4000000000000000'},
 '2': {'len': b'10', 'data': b'1234567890'}}
```

Modify the decoded message to send a response back. Change message type from '0200' to '0210'. Remove field 2 (PAN). And add field 39 (Response Code).

```
>>> decoded['t'] = '0210'
>>> decoded.pop('2', None)
'1234567890'
>>> decoded['39'] = '05'
```

Use iso8583.encode to encode updated ISO8583 message. It returns a raw ISO8583 message and a dictionary with encoded data.

```
>>> encoded_raw, encoded = iso8583.encode(decoded, spec)
>>> encoded_raw
bytearray(b'02100000000002000000005')
>>> pprint.pp(decoded)
{'t': '0210', 'p': '0000000002000000', '39': '05'}
>>> pprint.pp(encoded)
{'t': {'len': b'', 'data': b'0210'},
 'p': {'len': b'', 'data': b'0000000002000000'},
 '39': {'len': b'', 'data': b'05'}}
```

Pretty Print Messages

Use iso8583.pp to pretty print ISO8583 message.

```
>>> import iso8583
>>> from iso8583.specs import default_ascii as spec
>>> encoded_raw = b'02004000000000000000101234567890'
>>> decoded, encoded = iso8583.decode(encoded_raw, spec)
>>> iso8583.pp(decoded, spec)
t   Message Type                  : '0200'
p   Bitmap, Primary               : '4000000000000000'
2   Primary Account Number (PAN)  : '1234567890'
>>> iso8583.pp(encoded, spec)
t   Message Type                  : b'0200'
p   Bitmap, Primary               : b'4000000000000000'
2   Primary Account Number (PAN)  : b'10' b'1234567890'
```

Contribute

`iso8583` package is hosted on GitHub.

Feel free to fork and send contributions over.

1.2 How-To

1.2.1 Create Own/Proprietary Specifications

`iso8583` comes with specification samples in iso8583.specs.py. Feel free to copy sample specification and modify it to your needs.

Multiple specifications can be supported at the same time. Simply declare additional specification dictionary.

Refer to *`iso8583.specs`* for configuration details.

1.2.2 Create ISO8583 Message

`iso8583` converts a regular Python `dict` into a `bytearray`. This dictionary must consist of `str` keys and `str` values.

At the minimum it must have key `'t'` which is the message type (e.g. `'0200'`).

```
>>> import pprint
>>> import iso8583
>>> from iso8583.specs import default_ascii as spec
>>> decoded = {'t': '0200'}
>>> encoded_raw, encoded = iso8583.encode(decoded, spec)
>>> encoded_raw
bytearray(b'02000000000000000000')
>>> pprint.pp(encoded)
{'t': {'len': b'', 'data': b'0200'},
 'p': {'len': b'', 'data': b'0000000000000000'}}
>>> pprint.pp(decoded)
{'t': '0200', 'p': '0000000000000000'}
```

Note that primary bitmap was generated automatically.

An ISO8583 message without any fields is not very useful. To add fields to the message simply add `'2'-'128'` keys with `str` data and re-encode.

```
>>> decoded['2'] = 'cardholder PAN'
>>> decoded['3'] = '111111'
>>> decoded['21'] = '021'
>>>
>>> encoded_raw, encoded = iso8583.encode(decoded, spec)
>>> encoded_raw
bytearray(b'0200600008000000000014cardholder PAN111111021')
>>> pprint.pp(encoded)
{'t': {'len': b'', 'data': b'0200'},
 'p': {'len': b'', 'data': b'6000080000000000'},
 '2': {'len': b'14', 'data': b'cardholder PAN'},
 '3': {'len': b'', 'data': b'111111'},
 '21': {'len': b'', 'data': b'021'}}
>>> pprint.pp(decoded)
{'t': '0200',
 'p': '6000080000000000',
 '2': 'cardholder PAN',
 '3': '111111',
 '21': '021'}
```

Let's remove some fields and re-encode.

```
>>> decoded.pop('2', None)
'cardholder PAN'
>>> decoded.pop('3', None)
'111111'
>>>
>>> encoded_raw, encoded = iso8583.encode(decoded, spec)
>>> encoded_raw
bytearray(b'020000000800000000000021')
>>> pprint.pp(encoded)
{'t': {'len': b'', 'data': b'0200'},
 'p': {'len': b'', 'data': b'0000080000000000'},
 '21': {'len': b'', 'data': b'021'}}
>>> pprint.pp(decoded)
{'t': '0200', 'p': '0000080000000000', '21': '021'}
```

1.2.3 Add Secondary Bitmap

There is no need to explicitly add or remove secondary bitmap. It's auto generated when at least one `'65'-'128'` fields is present.

```
>>> import pprint
>>> import iso8583
>>> from iso8583.specs import default_ascii as spec
>>> decoded = {
...     't': '0200',
...     '102': '111111'}
>>> encoded_raw, encoded = iso8583.encode(decoded, spec)
>>> encoded_raw
bytearray(b'020080000000000000000000000004000000006111111')
```

(continues on next page)

(continued from previous page)

```
>>> pprint.pp(encoded)
{'t': {'len': b'', 'data': b'0200'},
 'p': {'len': b'', 'data': b'8000000000000000'},
 '1': {'len': b'', 'data': b'0000000004000000'},
 '102': {'len': b'06', 'data': b'111111'}}
>>> pprint.pp(decoded)
{'t': '0200', '102': '111111', 'p': '8000000000000000', '1': '0000000004000000'}
```

Even if secondary (or primary) bitmap is specified it's overwritten with correct value.

```
>>> decoded = {
...     't': '0200',
...     'p': 'spam',
...     '1': 'eggs',
...     '102': '111111'}
>>> encoded_raw, encoded = iso8583.encode(decoded, spec)
>>> encoded_raw
bytearray(b'02008000000000000000000000000400000006111111')
>>> pprint.pp(encoded)
{'t': {'len': b'', 'data': b'0200'},
 'p': {'len': b'', 'data': b'8000000000000000'},
 '1': {'len': b'', 'data': b'0000000004000000'},
 '102': {'len': b'06', 'data': b'111111'}}
>>> pprint.pp(decoded)
{'t': '0200', 'p': '8000000000000000', '102': '111111', '1': '0000000004000000'}
```

Secondary bitmap is removed if it's not required.

```
>>> decoded = {
...     't': '0200',
...     'p': 'spam',
...     '1': 'eggs',
...     '21': '051'}
>>> encoded_raw, encoded = iso8583.encode(decoded, spec)
>>> encoded_raw
bytearray(b'020000000800000000000051')
>>> pprint.pp(encoded)
{'t': {'len': b'', 'data': b'0200'},
 'p': {'len': b'', 'data': b'0000080000000000'},
 '21': {'len': b'', 'data': b'051'}}
>>> pprint.pp(decoded)
{'t': '0200', 'p': '0000080000000000', '21': '051'}
```

1.2.4 Check for Mandatory Fields

Many ISO8583 implementations need to check if all mandatory fields are received. It's easy to do this using a set of mandatory fields and checking if it's a subset of the received fields.

```
>>> import pprint
>>> import iso8583
>>> from iso8583.specs import default_ascii as spec
>>> encoded_raw = b'020080000800000000000000000004000000005106111111'
>>> decoded, encoded = iso8583.decode(encoded_raw, spec)
>>> pprint.pp(decoded)
{'t': '0200',
```

(continues on next page)

(continued from previous page)

```
 'p': '8000080000000000',
 '1': '0000000004000000',
 '21': '051',
 '102': '111111'}
>>> mandatory_fields = frozenset(['2', '21', '102'])
>>> mandatory_fields.issubset(decoded.keys())
False
>>> mandatory_fields = frozenset(['21', '102'])
>>> mandatory_fields.issubset(decoded.keys())
True
```

1.2.5 Convert to and from JSON

`iso8583` output is JSON compatible.

```
>>> import json
>>> import pprint
>>> import iso8583
>>> from iso8583.specs import default_ascii as spec
>>> encoded_raw = b'0200600008000000000014cardholder PAN111111021'
>>> decoded, encoded = iso8583.decode(encoded_raw, spec)
>>> pprint.pp(decoded)
{'t': '0200',
 'p': '6000080000000000',
 '2': 'cardholder PAN',
 '3': '111111',
 '21': '021'}
>>> decoded_json = json.dumps(decoded)
>>> decoded_json
'{"t": "0200", "p": "6000080000000000", "2": "cardholder PAN", "3": "111111", "21":
↪"021"}'
```

And back.

```
>>> encoded_raw, encoded = iso8583.encode(json.loads(decoded_json), spec)
>>> encoded_raw
bytearray(b'0200600008000000000014cardholder PAN111111021')
```

`iso8583.specs` specifications are also JSON compatible.

```
>>> import json
>>> import pprint
>>> import iso8583
>>> from iso8583.specs import default_ascii as spec
>>> spec_json = json.dumps(spec)
>>> decoded = {
...     't': '0200',
...     '2': 'PAN'}
>>> encoded_raw, encoded = iso8583.encode(decoded, json.loads(spec_json))
>>> encoded_raw
bytearray(b'0200400000000000000003PAN')
>>> pprint.pp(encoded)
{'t': {'len': b'', 'data': b'0200'},
 'p': {'len': b'', 'data': b'4000000000000000'},
 '2': {'len': b'03', 'data': b'PAN'}}
```

(continues on next page)

(continued from previous page)

```
>>> pprint.pp(decoded)
{'t': '0200', '2': 'PAN', 'p': '4000000000000000'}
```

1.2.6 Sending Binary Data

Sometimes data needs to be sent in binary. Let's add some EMV data as a hexchar string.

```
>>> import iso8583
>>> from iso8583.specs import default_ascii as spec
>>> decoded = {
...     't': '0200',
...     '55': '9F02060000000123455F340102'}
```

Let's define field 55 as an LLLVAR binary field with ascii length. Normally, this should be defined in the specification itself. But for demonstration purposes specification can be changed on the fly.

```
>>> spec['55']['data_enc'] = 'b'
>>> spec['55']['len_enc'] = 'ascii'
>>> spec['55']['len_type'] = 3
>>> spec['55']['max_len'] = 999
```

And encode the data.

```
>>> encoded_raw, encoded = iso8583.encode(decoded, spec)
>>> encoded_raw
bytearray(b'0200000000000000000200013\x9f\x02\x06\x00\x00\x00\x01#E_4\x01\x02')
```

1.2.7 Sending Binary Coded Decimal Data

Some ancient systems require data to be sent in BCD. Let's send an odd length PAN as an LLVAR BCD field with BCD length. Since, this field has to handle odd data, let's also define a PAD digit.

```
>>> import iso8583
>>> from iso8583.specs import default_ascii as spec
>>> decoded = {
...     't': '0200',
...     '2': '121200000002121'}
```

Let's define field 2 as an LLVAR BCD field with BCD length measured in nibbles. The field is padded on the right with a `'0'` because half a byte cannot be sent by itself.

```
>>> spec['2']['data_enc'] = 'b'
>>> spec['2']['len_enc'] = 'b'
>>> spec['2']['len_type'] = 1
>>> spec['2']['max_len'] = 19
>>> spec['2']['len_count'] = 'nibbles'
>>> spec['2']['right_pad'] = '0'
```

And encode the data.

```
>>> encoded_raw, encoded = iso8583.encode(decoded, spec)
>>> encoded_raw
bytearray(b'0200400000000000000\x15\x12\x12\x00\x00\x00\x02\x12\x10')
```

1.3 Specifications

An ISO8583 spec is a Python dictionary. It describes how each field needs to be encoded and decoded.

1.3.1 Fields

Supported fields:

- **h** - Message Header. If a specification does not have a header then set **max_len** to `0` to disable it.
- **t** - Message Type Identifier
- **p** - Primary Bitmap
- **1** - Secondary Bitmap
- **2** .. **128** - Regular fields

1.3.2 Mandatory Field Properties

Each field defines these properties:

- **data_enc** - field data encoding type. Set to:
  - `b` for binary data. For example, `ABCD` hex string is encoded as `\xAB\xCD` 2-byte value. This encoding type is also used for Binary-Coded Decimal (BCD) because BCD encoding is a subset of hex encoding. For example, `1234` numeric string is encoded as `\x12\x34` 2-byte BCD value.
  - Or any valid Python encoding. For example, `ascii` or `latin-1` for ASCII data and `cp500` or similar for EBCDIC data. For a list of built-in encodings, see Python standard encodings: https://docs.python.org/3/library/codecs.html#standard-encodings
- **len_enc** - field length encoding type. This is needed for some fields, such as ICC data, that could have binary data but ASCII length. This parameter does not affect fixed-length fields. Set to:
  - `b` for binary length. For example, 16 byte long field will have length encoded as `\x10`.
  - `bcd` for Binary-Coded Decimal (BCD) length. For example, 16 byte long field will have length encoded as `\x16`.
  - Or any valid Python encoding. For example, `ascii` or `latin-1` for ASCII length and `cp500` or similar for EBCDIC length. For a list of built-in encodings, see Python standard encodings: https://docs.python.org/3/library/codecs.html#standard-encodings
- **len_type** - field length type: fixed, LVAR, LLVAR, etc. Expressed as a number of bytes in field length. For example, ASCII LLVAR length takes up 2 bytes (`b'00' - b'99'`). BCD LLVAR length can take up only 1 byte (`b'\x00' - b'\x99'`). For binary lengths lenght type specifies the width of an unsigned integer. Therefore, **len_type** depends on the type of **len_enc** being used. BCD and binary **len_enc** can fit higher length ranges in fewer bytes.

len_type	**len_enc**		
	ASCII / EBCDIC	Binary-Coded Decimal	Binary
Fixed	`0`	`0`	`0`
LVAR	`1`	`1`	`1` (uint8)
LLVAR	`2`	`1`	`1` (uint8)
LLLVAR	`3`	`2`	`2` (uint16)
LLLLVAR	`4`	`2`	`2` (uint16)

- **max_len** - field maximum length in bytes or nibbles. For fixed fields **max_len** defines the length of the field.
- **desc** - field description that's printed in a pretty format. **desc** plays no role in encoding or decoding data. It's safe to omit it from the specifications. However, if omitted *iso8583.pp()* may or may not work as expected.

1.3.3 Optional Field Properties

Each field may define these additional properties as needed:

- **len_count** - specifies if field length is measured in bytes or nibbles (half bytes). This parameter affects **max_len**.
  - Use `bytes` (default) to measure field length in bytes.
  - Use `nibbles` to measure field length if nibbles (half bytes).
- **left_pad** - specifies pad character to be added/removed on the left side of an odd binary or BCD field without this character being included into field length. Valid pad character is `0-9` for BCD fields and `0-F` for binary fields.

  This option is used only when **data_enc** is set to `b` (binary/BCD) and **len_count** is set to `nibbles`. This option is meant for specifications that require odd length binary or BCD data.
- **right_pad** - same as **left_pad** but it pads on the right side. Specify either **left_pad** or **right_pad**. If both are specified at the same time then **left_pad** takes precedence.

1.3.4 Sample Field Specifications

Binary primary bitmap. Length encoding makes no difference, since the field has no length:

```
specification["p"] = {
    "data_enc": "b",
    "len_enc": "ascii",
    "len_type": 0,
    "max_len": 8,
    "desc": "Binary bitmap, e.g. \x12\x34\x56\x78\x90\xAB\xCD\xEF"
}
```

Hex string primary bitmap. Length encoding makes no difference, since the field has no length:

```
specification["p"] = {
    "data_enc": "ascii",
    "len_enc": "ascii",
    "len_type": 0,
    "max_len": 16,
    "desc": "Hex string bitmap, e.g 1234567890ABCDEF"
}
```

Field 2, a 10-byte binary or BCD fixed length field. Length encoding makes no difference, since the field has no length:

```
specification["2"] = {
    "data_enc": "b",
    "len_enc": "ascii",
    "len_type": 0,
    "max_len": 10,
    "desc": "Binary or BCD fixed field"
}
```

Field 3, an ASCII LLVAR field of maximum 20 bytes:

```
specification["3"] = {
    "data_enc": "ascii",
    "len_enc": "ascii",
    "len_type": 2,
    "max_len": 20,
    "desc": "ASCII LLVAR field"
}
```

Field 4, an EBCDIC LLLVAR field of maximum 150 bytes:

```
specification["4"] = {
    "data_enc": "cp500",
    "len_enc": "cp500",
    "len_type": 3,
    "max_len": 150,
    "desc": "EBCDIC LLLVAR field"
}
```

Field 5, a binary or BCD LLVAR field measured in nibbles and left-padded with 0. The field is maximum 20 nibbles:

```
specification["5"] = {
    "data_enc": "b",
    "len_enc": "bcd",
    "len_type": 1,
    "len_count": "nibbles",
    "left_pad": "0",
    "max_len": 20,
    "desc": "Binary or BCD LLVAR field measured in nibbles, e.g. \x03\x01\x11"
}
```

Field 6, a 3-nibble binary or BCD fixed field right-padded with 0. Length encoding makes no difference, since the field has no length:

```
specification["6"] = {
    "data_enc": "b",
    "len_enc": "ascii",
    "len_type": 0,
    "len_count": "nibbles",
    "right_pad": "0",
    "max_len": 3,
    "desc": "Binary or BCD fixed field measured in nibbles, e.g. \x11\x10"
}
```

Field 7, a 16-byte ascii field with binary length:

```
specification["2"] = {
    "data_enc": "ascii",
    "len_enc": "b",
    "len_type": 1,
    "max_len": 16,
    "desc": "ASCII field with binary length, e.g. \x101234567890123456"
}
```

1.3.5 Sample Message Specifications

ASCII/Binary

Bitmaps, MACs, PIN, and ICC data are in binary:

```
default = {
"h":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 0,   "desc
↪": "Message Header"},
"t":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 4,   "desc
↪": "Message Type"},
"p":   {"data_enc": "b",     "len_enc": "ascii", "len_type": 0, "max_len": 8,   "desc
↪": "Bitmap, Primary"},
"1":   {"data_enc": "b",     "len_enc": "ascii", "len_type": 0, "max_len": 8,   "desc
↪": "Bitmap, Secondary"},
"2":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 19,  "desc
↪": "Primary Account Number (PAN)"},
"3":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 6,   "desc
↪": "Processing Code"},
"4":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 12,  "desc
↪": "Amount, Transaction"},
"5":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 12,  "desc
↪": "Amount, Settlement"},
"6":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 12,  "desc
↪": "Amount, Cardholder Billing"},
"7":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 10,  "desc
↪": "Transmission Date and Time"},
"8":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 8,   "desc
↪": "Amount, Cardholder Billing Fee"},
"9":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 8,   "desc
↪": "Conversion Rate, Settlement"},
"10":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 8,   "desc
↪": "Conversion Rate, Cardholder Billing"},
"11":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 6,   "desc
↪": "System Trace Audit Number"},
"12":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 6,   "desc
↪": "Time, Local Transaction"},
"13":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 4,   "desc
↪": "Date, Local Transaction"},
"14":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 4,   "desc
↪": "Date, Expiration"},
"15":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 4,   "desc
↪": "Date, Settlement"},
"16":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 4,   "desc
↪": "Date, Conversion"},
"17":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 4,   "desc
↪": "Date, Capture"},
"18":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 4,   "desc
↪": "Merchant Type"},
"19":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Acquiring Institution Country Code"},
"20":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "PAN Country Code"},
"21":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Forwarding Institution Country Code"},
"22":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Point-of-Service Entry Mode"},
"23":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "PAN Sequence Number"},
"24":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Network International ID (NII)"},
```

(continues on next page)

(continued from previous page)

```
"25":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 2,   "desc
↪": "Point-of-Service Condition Code"},
"26":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 2,   "desc
↪": "Point-of-Service Capture Code"},
"27":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 1,   "desc
↪": "Authorizing ID Response Length"},
"28":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 9,   "desc
↪": "Amount, Transaction Fee"},
"29":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 9,   "desc
↪": "Amount, Settlement Fee"},
"30":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 9,   "desc
↪": "Amount, Transaction Processing Fee"},
"31":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 9,   "desc
↪": "Amount, Settlement Processing Fee"},
"32":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 11,  "desc
↪": "Acquiring Institution ID Code"},
"33":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 11,  "desc
↪": "Forwarding Institution ID Code"},
"34":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 28,  "desc
↪": "Primary Account Number, Extended"},
"35":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 37,  "desc
↪": "Track 2 Data"},
"36":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 104, "desc
↪": "Track 3 Data"},
"37":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 12,  "desc
↪": "Retrieval Reference Number"},
"38":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 6,   "desc
↪": "Authorization ID Response"},
"39":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 2,   "desc
↪": "Response Code"},
"40":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Service Restriction Code"},
"41":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 8,   "desc
↪": "Card Acceptor Terminal ID"},
"42":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 15,  "desc
↪": "Card Acceptor ID Code"},
"43":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 40,  "desc
↪": "Card Acceptor Name/Location"},
"44":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 25,  "desc
↪": "Additional Response Data"},
"45":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 76,  "desc
↪": "Track 1 Data"},
"46":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Additional Data - ISO"},
"47":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Additional Data - National"},
"48":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Additional Data - Private"},
"49":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Currency Code, Transaction"},
"50":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Currency Code, Settlement"},
"51":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Currency Code, Cardholder Billing"},
"52":  {"data_enc": "b",     "len_enc": "ascii", "len_type": 0, "max_len": 8,   "desc
↪": "PIN"},
"53":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 16,  "desc
↪": "Security-Related Control Information"},
```

(continues on next page)

(continued from previous page)

```
"54":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 240, "desc
↪": "Additional Amounts"},
"55":  {"data_enc": "b",     "len_enc": "ascii", "len_type": 3, "max_len": 255, "desc
↪": "ICC data"},
"56":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved ISO"},
"57":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved National"},
"58":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved National"},
"59":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved National"},
"60":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved National"},
"61":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved Private"},
"62":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved Private"},
"63":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved Private"},
"64":  {"data_enc": "b",     "len_enc": "ascii", "len_type": 0, "max_len": 8,   "desc
↪": "MAC"},
"65":  {"data_enc": "b",     "len_enc": "ascii", "len_type": 0, "max_len": 8,   "desc
↪": "Bitmap, Extended"},
"66":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 1,   "desc
↪": "Settlement Code"},
"67":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 2,   "desc
↪": "Extended Payment Code"},
"68":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Receiving Institution Country Code"},
"69":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Settlement Institution Country Code"},
"70":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Network Management Information Code"},
"71":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 4,   "desc
↪": "Message Number"},
"72":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 4,   "desc
↪": "Message Number, Last"},
"73":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 6,   "desc
↪": "Date, Action"},
"74":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 10,  "desc
↪": "Credits, Number"},
"75":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 10,  "desc
↪": "Credits, Reversal Number"},
"76":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 10,  "desc
↪": "Debits, Number"},
"77":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 10,  "desc
↪": "Debits, Reversal Number"},
"78":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 10,  "desc
↪": "Transfer, Number"},
"79":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 10,  "desc
↪": "Transfer, Reversal Number"},
"80":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 10,  "desc
↪": "Inquiries, Number"},
"81":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 10,  "desc
↪": "Authorizations, Number"},
"82":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 12,  "desc
↪": "Credits, Processing Fee Amount"},
```

(continues on next page)

(continued from previous page)

```
"83":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 12,  "desc
↪": "Credits, Transaction Fee Amount"},
"84":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 12,  "desc
↪": "Debits, Processing Fee Amount"},
"85":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 12,  "desc
↪": "Debits, Transaction Fee Amount"},
"86":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 16,  "desc
↪": "Credits, Amount"},
"87":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 16,  "desc
↪": "Credits, Reversal Amount"},
"88":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 16,  "desc
↪": "Debits, Amount"},
"89":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 16,  "desc
↪": "Debits, Reversal Amount"},
"90":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 42,  "desc
↪": "Original Data Elements"},
"91":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 1,   "desc
↪": "File Update Code"},
"92":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 2,   "desc
↪": "File Security Code"},
"93":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 5,   "desc
↪": "Response Indicator"},
"94":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 7,   "desc
↪": "Service Indicator"},
"95":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 42,  "desc
↪": "Replacement Amounts"},
"96":  {"data_enc": "b",     "len_enc": "ascii", "len_type": 0, "max_len": 8,   "desc
↪": "Message Security Code"},
"97":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 17,  "desc
↪": "Amount, Net Settlement"},
"98":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 25,  "desc
↪": "Payee"},
"99":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 11,  "desc
↪": "Settlement Institution ID Code"},
"100": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 11,  "desc
↪": "Receiving Institution ID Code"},
"101": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 17,  "desc
↪": "File Name"},
"102": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 28,  "desc
↪": "Account ID 1"},
"103": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 28,  "desc
↪": "Account ID 2"},
"104": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 100, "desc
↪": "Transaction Description"},
"105": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for ISO Use"},
"106": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for ISO Use"},
"107": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for ISO Use"},
"108": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for ISO Use"},
"109": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for ISO Use"},
"110": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for ISO Use"},
"111": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for ISO Use"},
```

(continues on next page)

(continued from previous page)

```
"112": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for National Use"},
"113": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for National Use"},
"114": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for National Use"},
"115": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for National Use"},
"116": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for National Use"},
"117": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for National Use"},
"118": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for National Use"},
"119": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for National Use"},
"120": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for Private Use"},
"121": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for Private Use"},
"122": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for Private Use"},
"123": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for Private Use"},
"124": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for Private Use"},
"125": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for Private Use"},
"126": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for Private Use"},
"127": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for Private Use"},
"128": {"data_enc": "b",     "len_enc": "ascii", "len_type": 0, "max_len": 8,   "desc
↪": "MAC"}
}
```

ASCII

All fields are in ASCII:

```
default_ascii = {
"h":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 0,   "desc
↪": "Message Header"},
"t":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 4,   "desc
↪": "Message Type"},
"p":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 16,  "desc
↪": "Bitmap, Primary"},
"1":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 16,  "desc
↪": "Bitmap, Secondary"},
"2":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 19,  "desc
↪": "Primary Account Number (PAN)"},
"3":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 6,   "desc
↪": "Processing Code"},
"4":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 12,  "desc
↪": "Amount, Transaction"},
```

(continues on next page)

(continued from previous page)

```
"5":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 12,  "desc
↪": "Amount, Settlement"},
"6":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 12,  "desc
↪": "Amount, Cardholder Billing"},
"7":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 10,  "desc
↪": "Transmission Date and Time"},
"8":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 8,   "desc
↪": "Amount, Cardholder Billing Fee"},
"9":   {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 8,   "desc
↪": "Conversion Rate, Settlement"},
"10":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 8,   "desc
↪": "Conversion Rate, Cardholder Billing"},
"11":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 6,   "desc
↪": "System Trace Audit Number"},
"12":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 6,   "desc
↪": "Time, Local Transaction"},
"13":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 4,   "desc
↪": "Date, Local Transaction"},
"14":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 4,   "desc
↪": "Date, Expiration"},
"15":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 4,   "desc
↪": "Date, Settlement"},
"16":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 4,   "desc
↪": "Date, Conversion"},
"17":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 4,   "desc
↪": "Date, Capture"},
"18":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 4,   "desc
↪": "Merchant Type"},
"19":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Acquiring Institution Country Code"},
"20":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "PAN Country Code"},
"21":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Forwarding Institution Country Code"},
"22":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Point-of-Service Entry Mode"},
"23":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "PAN Sequence Number"},
"24":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Network International ID (NII)"},
"25":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 2,   "desc
↪": "Point-of-Service Condition Code"},
"26":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 2,   "desc
↪": "Point-of-Service Capture Code"},
"27":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 1,   "desc
↪": "Authorizing ID Response Length"},
"28":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 9,   "desc
↪": "Amount, Transaction Fee"},
"29":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 9,   "desc
↪": "Amount, Settlement Fee"},
"30":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 9,   "desc
↪": "Amount, Transaction Processing Fee"},
"31":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 9,   "desc
↪": "Amount, Settlement Processing Fee"},
"32":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 11,  "desc
↪": "Acquiring Institution ID Code"},
"33":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 11,  "desc
↪": "Forwarding Institution ID Code"},
```

(continues on next page)

(continued from previous page)

```
"34":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 28,  "desc
↪": "Primary Account Number, Extended"},
"35":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 37,  "desc
↪": "Track 2 Data"},
"36":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 104, "desc
↪": "Track 3 Data"},
"37":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 12,  "desc
↪": "Retrieval Reference Number"},
"38":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 6,   "desc
↪": "Authorization ID Response"},
"39":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 2,   "desc
↪": "Response Code"},
"40":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Service Restriction Code"},
"41":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 8,   "desc
↪": "Card Acceptor Terminal ID"},
"42":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 15,  "desc
↪": "Card Acceptor ID Code"},
"43":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 40,  "desc
↪": "Card Acceptor Name/Location"},
"44":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 25,  "desc
↪": "Additional Response Data"},
"45":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 76,  "desc
↪": "Track 1 Data"},
"46":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Additional Data - ISO"},
"47":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Additional Data - National"},
"48":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Additional Data - Private"},
"49":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Currency Code, Transaction"},
"50":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Currency Code, Settlement"},
"51":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Currency Code, Cardholder Billing"},
"52":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 16,  "desc
↪": "PIN"},
"53":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 16,  "desc
↪": "Security-Related Control Information"},
"54":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 240, "desc
↪": "Additional Amounts"},
"55":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 510, "desc
↪": "ICC data"},
"56":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved ISO"},
"57":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved National"},
"58":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved National"},
"59":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved National"},
"60":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved National"},
"61":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved Private"},
"62":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved Private"},
```

(continues on next page)

(continued from previous page)

```
"63":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved Private"},
"64":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 16,  "desc
↪": "MAC"},
"65":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 16,  "desc
↪": "Bitmap, Extended"},
"66":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 1,   "desc
↪": "Settlement Code"},
"67":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 2,   "desc
↪": "Extended Payment Code"},
"68":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Receiving Institution Country Code"},
"69":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Settlement Institution Country Code"},
"70":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 3,   "desc
↪": "Network Management Information Code"},
"71":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 4,   "desc
↪": "Message Number"},
"72":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 4,   "desc
↪": "Message Number, Last"},
"73":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 6,   "desc
↪": "Date, Action"},
"74":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 10,  "desc
↪": "Credits, Number"},
"75":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 10,  "desc
↪": "Credits, Reversal Number"},
"76":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 10,  "desc
↪": "Debits, Number"},
"77":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 10,  "desc
↪": "Debits, Reversal Number"},
"78":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 10,  "desc
↪": "Transfer, Number"},
"79":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 10,  "desc
↪": "Transfer, Reversal Number"},
"80":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 10,  "desc
↪": "Inquiries, Number"},
"81":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 10,  "desc
↪": "Authorizations, Number"},
"82":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 12,  "desc
↪": "Credits, Processing Fee Amount"},
"83":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 12,  "desc
↪": "Credits, Transaction Fee Amount"},
"84":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 12,  "desc
↪": "Debits, Processing Fee Amount"},
"85":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 12,  "desc
↪": "Debits, Transaction Fee Amount"},
"86":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 16,  "desc
↪": "Credits, Amount"},
"87":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 16,  "desc
↪": "Credits, Reversal Amount"},
"88":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 16,  "desc
↪": "Debits, Amount"},
"89":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 16,  "desc
↪": "Debits, Reversal Amount"},
"90":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 42,  "desc
↪": "Original Data Elements"},
"91":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 1,   "desc
↪": "File Update Code"},
```

(continues on next page)

(continued from previous page)

```
"92":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 2,   "desc
↪": "File Security Code"},
"93":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 5,   "desc
↪": "Response Indicator"},
"94":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 7,   "desc
↪": "Service Indicator"},
"95":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 42,  "desc
↪": "Replacement Amounts"},
"96":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 16,  "desc
↪": "Message Security Code"},
"97":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 17,  "desc
↪": "Amount, Net Settlement"},
"98":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 25,  "desc
↪": "Payee"},
"99":  {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 11,  "desc
↪": "Settlement Institution ID Code"},
"100": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 11,  "desc
↪": "Receiving Institution ID Code"},
"101": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 17,  "desc
↪": "File Name"},
"102": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 28,  "desc
↪": "Account ID 1"},
"103": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 2, "max_len": 28,  "desc
↪": "Account ID 2"},
"104": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 100, "desc
↪": "Transaction Description"},
"105": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for ISO Use"},
"106": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for ISO Use"},
"107": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for ISO Use"},
"108": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for ISO Use"},
"109": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for ISO Use"},
"110": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for ISO Use"},
"111": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for ISO Use"},
"112": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for National Use"},
"113": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for National Use"},
"114": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for National Use"},
"115": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for National Use"},
"116": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for National Use"},
"117": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for National Use"},
"118": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for National Use"},
"119": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for National Use"},
"120": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for Private Use"},
```

(continues on next page)

(continued from previous page)

```
"121": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for Private Use"},
"122": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for Private Use"},
"123": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for Private Use"},
"124": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for Private Use"},
"125": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for Private Use"},
"126": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for Private Use"},
"127": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 3, "max_len": 999, "desc
↪": "Reserved for Private Use"},
"128": {"data_enc": "ascii", "len_enc": "ascii", "len_type": 0, "max_len": 16,  "desc
↪": "MAC"}
}
```

1.4 Functions

1.4.1 Main Functions

iso8583.**decode**(*s: Union[bytes, bytearray], spec: Mapping[str, Mapping[str, Any]]*) → Tuple[Dict[str, str], Dict[str, Dict[str, bytes]]]

Deserialize a bytes or bytearray instance containing ISO8583 data to a Python dict.

Parameters

- **s** (*bytes or bytearray*) – Encoded ISO8583 data
- **spec** (*dict*) – A Python dict defining ISO8583 specification. See *iso8583.specs* module for examples.

Returns

- **doc_dec** (*dict*) – Dict containing decoded ISO8583 data
- **doc_enc** (*dict*) – Dict containing encoded ISO8583 data

Raises

- *DecodeError* – An error decoding ISO8583 bytearray
- TypeError – *s* must be a bytes or bytearray instance

Examples

```
>>> import pprint
>>> import iso8583
>>> from iso8583.specs import default_ascii as spec
>>> s = b"0200401010000000000016123456789012345612345611"
>>> doc_dec, doc_enc = iso8583.decode(s, spec)
>>> pprint.pprint(doc_dec)
{'12': '123456',
 '2': '1234567890123456',
```

(continues on next page)

(continued from previous page)

```
 '20': '111',
 'p': '4010100000000000',
 't': '0200'}
```

iso8583.**encode**(*doc_dec: MutableMapping[str, str], spec: Mapping[str, Mapping[str, Any]]*) → Tuple[bytearray, Dict[str, Dict[str, bytes]]]

Serialize Python dict containing ISO8583 data to a bytearray.

Parameters

- **doc_dec** (*dict*) – Dict containing decoded ISO8583 data
- **spec** (*dict*) – A Python dict defining ISO8583 specification. See *iso8583.specs* module for examples.

Returns

- **s** (*bytearray*) – Encoded ISO8583 data
- **doc_enc** (*dict*) – Dict containing encoded ISO8583 data

Raises

- *EncodeError* – An error encoding ISO8583 bytearray
- TypeError – *doc_dec* must be a dict instance

Examples

```
>>> import iso8583
>>> from iso8583.specs import default_ascii as spec
>>> doc_dec = {
...     't': '0210',
...     '3': '111111',
...     '39': '05'}
>>> s, doc_enc = iso8583.encode(doc_dec, spec)
>>> s
bytearray(b'02102000000002000000111111105')
```

1.4.2 Exceptions

exception iso8583.**DecodeError**(*msg: str, s: Union[bytes, bytearray], doc_dec: Dict[str, str], doc_enc: Dict[str, Dict[str, bytes]], pos: int, field: str*)

Subclass of ValueError that describes ISO8583 decoding error.

Variables

- **msg** (*str*) – The unformatted error message
- **s** (*bytes or bytearray*) – The ISO8583 bytes instance being parsed
- **doc_dec** (*dict*) – Dict containing partially decoded ISO8583 data
- **doc_enc** (*dict*) – Dict containing partially encoded ISO8583 data
- **pos** (*int*) – The start index where ISO8583 bytes data failed parsing
- **field** (*str*) – The ISO8583 field where parsing failed

exception iso8583.**EncodeError**(*msg: str, doc_dec: MutableMapping[str, str], doc_enc: Dict[str, Dict[str, bytes]], field: str*)

Subclass of ValueError that describes ISO8583 encoding error.

Variables

- **msg** (*str*) – The unformatted error message
- **doc_dec** (*dict*) – Dict containing decoded ISO8583 data being encoded
- **doc_enc** (*dict*) – Dict containing partially encoded ISO8583 data
- **field** (*str*) – The ISO8583 field where parsing failed

1.4.3 Helper Functions

iso8583.**pp**(*doc: Union[Mapping[str, str], Mapping[str, Mapping[str, bytes]]], spec: Mapping[str, Mapping[str, Any]], desc_width: int = 30, stream: Optional[TextIO] = None, line_width: int = 80*) → None

Pretty Print Python dict containing ISO8583 data.

Parameters

- **doc** (*dict*) – Dict containing ISO8583 data
- **spec** (*dict*) – A Python dict defining ISO8583 specification. See iso8583.specs module for examples.
- **desc_width** (*int, optional*) – Field description width (default 30). Specify 0 to print no descriptions.
- **stream** (*stream, optional*) – An output stream. The only method used on the stream object is the file protocol's write() method. If not specified, the *iso8583.pp()* adopts *sys.stdout*.
- **line_width** (*int, optional*) – Attempted maximum width of output line (default 80).

Notes

For the most correct information to be displayed by *iso8583.pp()* it's recommended to call it after *iso8583.encode()* or *iso8583.decode()*.

Examples

```
>>> import iso8583
>>> from iso8583.specs import default_ascii as spec
>>> s = b"0200401010000000000016123456789012345612345684O"
>>> doc_dec, doc_enc = iso8583.decode(s, spec)
>>> iso8583.pp(doc_dec, spec)
t   Message Type                  : '0200'
p   Bitmap, Primary               : '4010100000000000'
2   Primary Account Number (PAN)  : '1234567890123456'
12  Time, Local Transaction       : '123456'
20  PAN Country Code              : '840'
```

1.5 Change Log

1.5.1 3.0.0 - 2023-06-05

- Add support for binary field length
- Drop support for Python 3.6
- Migration from 2.x.x: - 2.x.x used *len_enc* set to *b* and *bcd* to represent BCD length encoding. - 3.x.x changed that where *len_enc* set to *b* represents binary length encoding. - To migrate from 2.x.x update *len_enc* in all specificatations from *b* to *bcd*.

1.5.2 2.2.0 - 2022-01-30

- Provide friendlier error messages when failing to encode/decode field, field length, and bitmap.
- Clarify Binary-coded decimal field length configuration. Added `bcd` value to `len_enc` which is the same as the existing `b` value. Both mean that the length is to be encoded as BCD.

1.5.3 2.1.0 - 2020-12-24

- Added support for fields measured in nibbles (half-bytes).

1.5.4 2.0.2 - 2020-11-20

- Fixed issue #4. Encode secondary bitmap in upper case.

1.5.5 2.0.1 - 2020-08-22

- Include inline type information into the distribution according to PEP 561.
- Address remaining type hint issues.

1.5.6 2.0.0 - 2020-02-21

Backwards incompatible:

- *iso8583.decode()* is updated not to produce bitmap key `'bm'`.
- *iso8583.encode()* is updated not to expect bitmap key `'bm'` to define fields to encode. The decision on what fields to encode is based on numeric fields in the range of `'1'-'128'` present in the decoded dictionary.
- `iso8583.add_field()` and `iso8583.del_field()` are removed. With the removal of bitmap set `'bm'` default Python dictionary methods are enough.
- *iso8583.encode()* now removes secondary bitmap key `'1'` from the decoded dictionary if no `'65'-'128'` fields are present.
- *iso8583.pp()* function signature changed. The first parameter is renamed from `doc_dec` to `doc`.

Other changes:

- *iso8583.pp()* handles both encoded and decoded dictionary output.

- *iso8583.pp()* handles output folding. The defaul line width is set to 80. Line width can be configured using new `line_width` parameter.

1.5.7 1.0.2 - 2020-01-11

- Optional proprietary header can now be parsed using standard field settings
- Documentation improvements

1.5.8 1.0.1 - 2019-11-11

- *iso8583.decode()* and *iso8583.decode()* now return a tuple
- *iso8583.decode()* returns a tuple of decoded dict instance and encoded dict instance
- *iso8583.encode()* returns a tuple of encoded bytes instance and encoded dict instance
- Encoded and decoded dict instance keys are now all strings
- Specification keys are now all strings

1.5.9 1.0.0 - 2019-11-04

Initial release.

Python Module Index

i

iso8583.specs, 10

Index

D

decode() (*in module iso8583*), 22
DecodeError, 23

E

encode() (*in module iso8583*), 23
EncodeError, 23

I

iso8583.specs (*module*), 10

P

pp() (*in module iso8583*), 24