python-fints Documentation

Release 4.0

Raphael Michel

Aug 22, 2023

Contents

This is a pure-python implementation of FinTS (formerly known as HBCI), a online-banking protocol commonly supported by German banks.

CHAPTER 1

Library user documentation content

1.1 Getting started

1.1.1 Register for a product ID

As of September 14th, 2019, all FinTS client programs need to be registered with the ZKA. You need to fill out a PDF form and will be assigned a product ID that you can pass to this library. It can take up to two weeks for the product ID to be assigned.

The reason for this requirement is compliance with the European Unions 2nd Payment Services Directive (PSD2) which mandates that end-users can transparently see which applications are accessing their bank account.

You cna find more information as well as the registration form on the ZKA Website (only available in German).

1.1.2 Start coding

First of all, you need to install the library:

```
$ pip3 install fints
```

Then, you can initialize a FinTS client by providing your bank's BLZ, your username and PIN as well as the HBCI endpoint of your bank. Logging in with a signature file or chip card is currently not supported. For example:

```
import logging
from datetime import date
import getpass
from fints.client import FinTS3PinTanClient

logging.basicConfig(level=logging.DEBUG)
f = FinTS3PinTanClient(
    '123456789',  # Your bank's BLZ
    'myusername',  # Your login name
    getpass.getpass('PIN:'),  # Your banking PIN
```

(continues on next page)

(continued from previous page)

```
    'https://hbci-pintan.gad.de/cgi-bin/hbciservlet',
    product_id='Your product ID'  # see above
)
```

Since the implementation of PSD2, you will in almost all cases need to be ready to deal with TANs. For a quick start, we included a minimal command-line utility to help choose a TAN method:

```
from fints.utils import minimal_interactive_cli_bootstrap
minimal_interactive_cli_bootstrap(f)
```

You can then open up a real communication dialog to the bank with a `with` statement and issue commands: commands using the client instance:

```
with f:
    # Since PSD2, a TAN might be needed for dialog initialization. Let's check if
↪there is one required
    if f.init_tan_response:
        print("A TAN is required", f.init_tan_response.challenge)
        tan = input('Please enter TAN:')
        f.send_tan(f.init_tan_response, tan)

    # Fetch accounts
    accounts = f.get_sepa_accounts()
```

Go on to the next pages to find out what commands are supported!

1.2 Reading operations

Note: Starting from version 3, **all of the methods on this page** can return a `NeedTANResponse` instead of actual data if your bank requires a TAN. You should then enter a TAN, read our chapter *Working with TANs* to find out more.

1.2.1 Fetching your bank accounts

The most simple method allows you to get all bank accounts that your user has access to:

class fints.client.**FinTS3Client**(*bank_identifier*, *user_id*, *customer_id=None*, *from_data: bytes = None*, *product_id=None*, *product_version='4.0.0'*, *mode=<FinTSClientMode.INTERACTIVE: 'interactive'>*)

get_sepa_accounts()

Returns a list of SEPA accounts

Returns List of SEPAAccount objects.

This method will return a list of named tuples of the following type:

class fints.models.**SEPAAccount**(*iban*, *bic*, *accountnumber*, *subaccount*, *blz*)

You will need this account object for many further operations to show which account you want to operate on.

1.2.2 Fetching bank information

During the first interaction with the bank some meta information about the bank and your user is transmitted from the bank.

class fints.client.**FinTS3Client**(*bank_identifier*, *user_id*, *customer_id=None*, *from_data: bytes = None*, *product_id=None*, *product_version='4.0.0'*, *mode=<FinTSClientMode.INTERACTIVE: 'interactive'>*)

get_information()

Return information about the connected bank.

Note: Can only be filled after the first communication with the bank. If in doubt, use a construction like:

```
f = FinTS3Client(...)
with f:
    info = f.get_information()
```

Returns a nested dictionary:

```
bank:
    name: Bank Name
    supported_operations: dict(FinTSOperations -> boolean)
    supported_formats: dict(FinTSOperation -> [
↪'urn:iso:std:iso:20022:tech:xsd:pain.001.003.03', ...])
    supported_sepa_formats: ['urn:iso:std:iso:20022:tech:xsd:pain.001.003.03',
↪ ...]
accounts:
    - iban: IBAN
      account_number: Account Number
      subaccount_number: Sub-Account Number
      bank_identifier: fints.formals.BankIdentifier(...)
      customer_id: Customer ID
      type: Account type
      currency: Currency
      owner_name: ['Owner Name 1', 'Owner Name 2 (optional)']
      product_name: Account product name
      supported_operations: dict(FinTSOperations -> boolean)
    - ...
```

1.2.3 Fetching account balances

You can fetch the current balance of an account with the `get_balance` operation.

class fints.client.**FinTS3Client**(*bank_identifier*, *user_id*, *customer_id=None*, *from_data: bytes = None*, *product_id=None*, *product_version='4.0.0'*, *mode=<FinTSClientMode.INTERACTIVE: 'interactive'>*)

get_balance(*account: fints.models.SEPAAccount*)

Fetches an accounts current balance.

Parameters **account** – SEPA account to fetch the balance

Returns A mt940.models.Balance object

This method will return a list of `Balance` objects from the `mt-940` library. You can find more information in their documentation.

1.2.4 Reading account transactions

You can fetch the banking statement of an account within a certain timeframe with the `get_transactions` operation.

class fints.client.**FinTS3Client**(*bank_identifier*, *user_id*, *customer_id=None*, *from_data: bytes = None*, *product_id=None*, *product_version='4.0.0'*, *mode=<FinTSClientMode.INTERACTIVE: 'interactive'>*)

get_transactions(*account: fints.models.SEPAAccount*, *start_date: datetime.date = None*, *end_date: datetime.date = None*)

Fetches the list of transactions of a bank account in a certain timeframe.

Parameters

- **account** – SEPA
- **start_date** – First day to fetch
- **end_date** – Last day to fetch

Returns A list of mt940.models.Transaction objects

get_transactions_xml(*account: fints.models.SEPAAccount*, *start_date: datetime.date = None*, *end_date: datetime.date = None*) → list

Fetches the list of transactions of a bank account in a certain timeframe as camt.052.001.02 XML files. Returns both booked and pending transactions.

Parameters

- **account** – SEPA
- **start_date** – First day to fetch
- **end_date** – Last day to fetch

Returns Two lists of bytestrings containing XML documents, possibly empty: first one for booked transactions, second for pending transactions

This method will return a list of `Transaction` objects from the `mt-940` library. You can find more information in their documentation.

1.2.5 Fetching holdings

You can fetch the holdings of an account with the `get_holdings` method:

class fints.client.**FinTS3Client**(*bank_identifier*, *user_id*, *customer_id=None*, *from_data: bytes = None*, *product_id=None*, *product_version='4.0.0'*, *mode=<FinTSClientMode.INTERACTIVE: 'interactive'>*)

get_holdings(*account: fints.models.SEPAAccount*)

Retrieve holdings of an account.

Parameters **account** – SEPAAccount to retrieve holdings for.

Returns List of Holding objects

This method will return a list of `Holding` objects:

class fints.models.**Holding**(*ISIN*, *name*, *market_value*, *value_symbol*, *valuation_date*, *pieces*, *total_value*, *acquisitionprice*)

1.3 The client object

1.3.1 Storing and restoring client state

The `FinTS3Client` object keeps some internal state that's beneficial to keep across invocations. This includes

- A system identifier that uniquely identifies this particular FinTS endpoint
- The Bank Parameter Data (BPD) with information about the bank and its advertised capabilities
- The User Parameter Data (UPD) with information about the user account and allowed actions

class fints.client.**FinTS3Client**(*bank_identifier*, *user_id*, *customer_id=None*, *from_data: bytes = None*, *product_id=None*, *product_version='4.0.0'*, *mode=<FinTSClientMode.INTERACTIVE: 'interactive'>*)

deconstruct(*including_private: bool = False*) → bytes

Return state of this FinTSClient instance as an opaque datablob. You should not use this object after calling this method.

Information about the connection is implicitly retrieved from the bank and cached in the FinTSClient. This includes: system identifier, bank parameter data, user parameter data. It's not strictly required to retain this information across sessions, but beneficial. If possible, an API user SHOULD use this method to serialize the client instance before destroying it, and provide the serialized data next time an instance is constructed.

Parameter *including_private* should be set to True, if the storage is sufficiently secure (with regards to confidentiality) to include private data, specifically, account numbers and names. Most often this is the case.

Note: No connection information is stored in the datablob, neither is the PIN.

set_data(*blob: bytes*)

Restore a datablob created with deconstruct().

You should only call this method once, and only immediately after constructing the object and before calling any other method or functionality (e.g. __enter__()). For convenience, you can pass the *from_data* parameter to __init__().

Using the `deconstruct()`/`set_data()` facility is purely optional for reading operations, but may speed up the process because the BPD/UPD can be cached and need not be transmitted again.

It may be required to use the facility for transaction operations if both parts of a two-step transaction cannot be completed with the same `FinTS3Client` object.

The `deconstruct()` parameter *include_private* (defaults to *False*) enables including the User Parameter Data in the datablob. Set this to *True* if you can sufficiently ensure the privacy of the returned datablob (mostly: user name and account numbers).

If your system manages multiple users/identity contexts, you SHOULD keep distinct datablobs per user or context.

You SHOULD NOT call any other methods on the `FinTS3Client` object after calling `deconstruct()`.

1.3.2 Keeping the dialog open

All FinTS operations happen in the context of a so-called "dialog". The simple reading operations of this library will automatically open and close the dialog when necessary, but each opening and each closing takes one FinTS roundtrip.

For the case where multiple operations are to be performed one after the other you can indicate to the library that you want to open a standing dialog and keep it open explicitly by entering the `FinTS3Client` as a context handler.

This can, and should be, complemented with the client state facility as follows:

```
datablob = ... # get from backend storage, or set to None
client = FinTS3PinTanClient(..., from_data=datablob)

with client:
    accounts = client.get_sepa_accounts()
    balance = client.get_balance(accounts[0])
    transactions = client.get_transactions(accounts[0])

datablob = client.deconstruct()
# Store datablob to backend storage
```

For transactions involving TANs it may be required by the bank to issue both steps for one transaction within the same dialog. In this case it's mandatory to use a standing dialog, because otherwise each step would be issued in its own, implicit, dialog.

1.3.3 Storing and restoring dialog state

class fints.client.**FinTS3Client**(*bank_identifier*, *user_id*, *customer_id=None*, *from_data: bytes = None*, *product_id=None*, *product_version='4.0.0'*, *mode=<FinTSClientMode.INTERACTIVE: 'interactive'>*)

pause_dialog()

Pause a standing dialog and return the saved dialog state.

Sometimes, for example in a web app, it's not possible to keep a context open during user input. In some cases, though, it's required to send a response within the same dialog that issued the original task (f.e. TAN with TANTimeDialogAssociation.NOT_ALLOWED). This method freezes the current standing dialog (started with FinTS3Client.__enter__()) and returns the frozen state.

Commands MUST NOT be issued in the dialog after calling this method.

MUST be used in conjunction with deconstruct()/set_data().

Caller SHOULD ensure that the dialog is resumed (and properly ended) within a reasonable amount of time.

Example

```
client = FinTS3PinTanClient(..., from_data=None)
with client:
    challenge = client.sepa_transfer(...)

    dialog_data = client.pause_dialog()

    # dialog is now frozen, no new commands may be issued
    # exiting the context does not end the dialog

client_data = client.deconstruct()

# Store dialog_data and client_data out-of-band somewhere
# ... Some time passes ...
# Later, possibly in a different process, restore the state

client = FinTS3PinTanClient(..., from_data=client_data)
with client.resume_dialog(dialog_data):
    client.send_tan(...)
```

(continues on next page)

(continued from previous page)

```
    # Exiting the context here ends the dialog, unless frozen with pause_
↪dialog() again.
```

resume_dialog(*dialog_data*)

1.4 Working with TANs

Many operations in FinTS will require a form of two-step authentication, called TANs. TANs are mostly required for operations that move money or change details of a bank account. TANs can be generated with a multitude of methods, including paper lists, smartcard readers, SMS messages, and smartphone apps.

1.4.1 TAN methods

Before doing any operations involving TANs, you should get a list of supported TAN mechanisms:

```
mechanisms = client.get_tan_mechanisms()
```

The returned dictionary maps identifiers (generally: three-digit numerals) to instances of a *TwoStepParametersCommon()* subclass with varying fields, depending on the version of the two-step process and the bank.

The *name* field of these objects provides a user-friendly name of the TAN mechanism that you can display to the user to choose from. To select a TAN mechanism, you can use `set_tan_mechanism()`, which takes the identifier used as key in the `get_tan_mechanisms()` return value.

If the `description_required` attribute for the TAN mechanism is *MUST*, you will need to get a list of TAN media with `get_tan_media()` and select the appropriate one with `set_tan_medium()`.

Have a look at the source code of `minimal_interactive_cli_bootstrap()` for an example on how to ask the user for these properties.

You may not change the active TAN mechanism or TAN medium within a standing dialog (see *Storing and restoring dialog state*).

The selection of the active TAN mechanism/medium is stored with the persistent client data (see *Storing and restoring client state*).

class fints.client.**FinTS3PinTanClient**(*bank_identifier*, *user_id*, *pin*, *server*, *customer_id=None*, **args*, ***kwargs*)

get_current_tan_mechanism()

get_tan_mechanisms()
Get the available TAN mechanisms.

Note: Only checks for HITANS versions listed in IMPLEMENTED_HKTAN_VERSIONS.

Returns Dictionary of security_function: TwoStepParameters objects.

get_tan_media(*media_type=fints.formals.TANMediaType2.ALL*, *media_class=fints.formals.TANMediaClass4.ALL*)
Get information about TAN lists/generators.

Returns tuple of fints.formals.TANUsageOption and a list of fints.formals.TANMedia4 or fints.formals.TANMedia5 objects.

set_tan_mechanism(*security_function*)

set_tan_medium(*tan_medium*)

1.4.2 TAN challenges

When you try to perform an operation that requires a TAN to proceed, you will receive an object containing the bank's challenge (and some internal data to continue the operation once the TAN has been processed):

class fints.client.**NeedTANResponse**(*command_seg*, *tan_request*, *resume_method=None*, *tan_request_structured=False*)

challenge = None
Textual challenge to be displayed to the user

challenge_hhduc = None
HHD_UC challenge to be transmitted to the TAN generator

challenge_html = None
HTML-safe challenge text, possibly with formatting

challenge_matrix = None
Matrix code challenge: tuple(mime_type, data)

challenge_raw = None
Raw challenge as received by the bank

get_data() → bytes
Return a compressed datablob representing this object.

To restore the object, use *fints.client.NeedRetryResponse.from_data()*.

The `challenge` attribute will contain human-readable instructions on how to proceed.

The `challenge_html` attribute will possibly contain a nicer, formatted, HTML version of the challenge text that you should prefer if your primary interface can render HTML. The contents are guaranteed to be proper and clean (by using the *bleach* library): They can be used with *mark_safe* in Django.

The `challenge_hhduc` attribute will contain the challenge to be used with a TAN generator device using the Hand Held Device Unidirectional Coupling specification (such as a Flicker-Code).

1.4.3 Flicker-Code / optiTAN

If you want to use chipTAN with an optical TAN device, we provide utilities to print the flicker code on a unix terminal. Just pass the `challenge_hhd_uc` value to this method:

fints.hhd.flicker.**terminal_flicker_unix**(*code*, *field_width=3*, *space_width=3*, *height=1*, *clear=False*, *wait=0.05*)
Re-encodes a flicker code and prints it on a unix terminal.

Parameters

- **code** – Challenge value
- **field_width** – Width of fields in characters (default: 3).
- **space_width** – Width of spaces in characters (default: 3).
- **height** – Height of fields in characters (default: 1).
- **clear** – Clear terminal after every line (default: `False`).

- **wait** – Waiting interval between lines (default: 0.05).

You should probably catch for `KeyboardInterrupts` to allow the user to abort the displaying and to continue with the TAN:

```
try:
    terminal_flicker_unix(result.challenge_hhduc)
except KeyboardInterrupt:
    pass
```

1.4.4 photoTAN

If you want to use photoTAN, use the `challenge_matrix` attribute to access the image file, e.g. by writing it to a file:

```
with open("tan.png", "wb") as writer:
   writer.write(result.challenge_matrix[1])
   writer.close()
```

1.4.5 Sending the TAN

Once obtained the TAN, you can send it with the `send_tan` client method:

class fints.client.**FinTS3PinTanClient**(*bank_identifier*, *user_id*, *pin*, *server*, *customer_id=None*, **args*, ***kwargs*)

send_tan(*challenge: fints.client.NeedTANResponse*, *tan: str*)

Sends a TAN to confirm a pending operation.

Parameters

- **challenge** – NeedTANResponse to respond to
- **tan** – TAN value

Returns Currently no response

For example:

```
tan = input('Please enter the TAN code: ')
result = client.send_tan(result, tan)
```

1.4.6 Storing and restoring TAN state

The *get_data()* method and *from_data()* factory method can be used to store and restore a TAN state object between steps.

class fints.client.**NeedRetryResponse**

Base class for Responses that need the operation to be externally retried.

A concrete subclass of this class is returned, if an operation cannot be completed and needs a retry/completion. Typical (and only) example: Requiring a TAN to be provided.

classmethod from_data(*blob*)

Restore an object instance from a compressed datablob.

Returns an instance of a concrete subclass.

You SHOULD use this facility together with the client and dialog state restoration facilities:

Listing 1: First step

```
client = FinTS3PinTanClient(...)
# Optionally: choose a tan mechanism with
# client.set_tan_mechanism(...)

with client:
    response = client.sepa_transfer(...)

    dialog_data = client.pause_dialog()
client_data = client.deconstruct()
tan_data = response.get_data()
```

Listing 2: Second step

```
tan_request = NeedRetryResponse.from_data(tan_data)
print("TAN request: {}".format(tan_request.challenge))
tan = input('Enter TAN: ')
```

Listing 3: Third step

```
tan_request = NeedRetryResponse.from_data(tan_data)
client = FinTS3PinTanClient(..., from_data=client_data)
with client.resume_dialog(dialog_data):
    response = client.send_tan(tan_request, tan)

print(response.status)
print(response.responses)
```

1.4.7 Reference

class fints.formals.**TwoStepParameters2**(**args*, ***kwargs*)

name
Name des Zwei-Schritt-Verfahrens

Type str

max_length_input
Maximale Länge des Eingabewertes im Zwei-Schritt-Verfahren

Type int

allowed_format
Erlaubtes Format im Zwei-Schritt-Verfahren

Type *fints.formals.AllowedFormat*

text_return_value
Text zur Belegung des Rückgabewertes im Zwei-Schritt-Verfahren

Type str

max_length_return_value
Maximale Länge des Rückgabewertes im Zwei-Schritt-Verfahren

Type int

number_of_supported_lists
Anzahl unterstützter aktiver TAN-Listen

Type int

multiple_tans_allowed
Mehrfach-TAN erlaubt

Type bool

tan_time_dialog_association
TAN Zeit- und Dialogbezug

Type *fints.formals.TANTimeDialogAssociation*

tan_list_number_required
TAN-Listennummer erforderlich

Type *fints.formals.TANListNumberRequired*

cancel_allowed
Auftragsstorno erlaubt

Type bool

challenge_class_required
Challenge-Klasse erforderlich

Type bool

challenge_value_required
Challenge-Betrag erforderlich

Type bool

VERSION
TAN mechanism version

class fints.formals.**TwoStepParameters3**(**args*, ***kwargs*)

name
Name des Zwei-Schritt-Verfahrens

Type str

max_length_input
Maximale Länge des Eingabewertes im Zwei-Schritt-Verfahren

Type int

allowed_format
Erlaubtes Format im Zwei-Schritt-Verfahren

Type *fints.formals.AllowedFormat*

text_return_value
Text zur Belegung des Rückgabewertes im Zwei-Schritt-Verfahren

Type str

max_length_return_value
Maximale Länge des Rückgabewertes im Zwei-Schritt-Verfahren

Type int

number_of_supported_lists
Anzahl unterstützter aktiver TAN-Listen

Type int

multiple_tans_allowed
Mehrfach-TAN erlaubt

Type bool

tan_time_dialog_association
TAN Zeit- und Dialogbezug

Type *fints.formals.TANTimeDialogAssociation*

tan_list_number_required
TAN-Listennummer erforderlich

Type *fints.formals.TANListNumberRequired*

cancel_allowed
Auftragsstorno erlaubt

Type bool

challenge_class_required
Challenge-Klasse erforderlich

Type bool

challenge_value_required
Challenge-Betrag erforderlich

Type bool

initialization_mode
Initialisierungsmodus

Type *fints.formals.InitializationMode*

description_required
Bezeichnung des TAN-Medium erforderlich

Type *fints.formals.DescriptionRequired*

supported_media_number
Anzahl unterstützter aktiver TAN-Medien

Type int

VERSION
TAN mechanism version

class fints.formals.**TwoStepParameters5**(**args*, ***kwargs*)

zka_id
ZKA TAN-Verfahren

Type str

zka_version
Version ZKA TAN-Verfahren

Type str

name
Name des Zwei-Schritt-Verfahrens

Type str

max_length_input
Maximale Länge des Eingabewertes im Zwei-Schritt-Verfahren

Type int

allowed_format
Erlaubtes Format im Zwei-Schritt-Verfahren

Type *fints.formals.AllowedFormat*

text_return_value
Text zur Belegung des Rückgabewertes im Zwei-Schritt-Verfahren

Type str

max_length_return_value
Maximale Länge des Rückgabewertes im Zwei-Schritt-Verfahren

Type int

number_of_supported_lists
Anzahl unterstützter aktiver TAN-Listen

Type int

multiple_tans_allowed
Mehrfach-TAN erlaubt

Type bool

tan_time_dialog_association
TAN Zeit- und Dialogbezug

Type *fints.formals.TANTimeDialogAssociation*

tan_list_number_required
TAN-Listennummer erforderlich

Type *fints.formals.TANListNumberRequired*

cancel_allowed
Auftragsstorno erlaubt

Type bool

sms_charge_account_required
SMS-Abbuchungskonto erforderlich

Type *fints.formals.SMSChargeAccountRequired*

principal_account_required
Auftraggeberkonto erforderlich

Type *fints.formals.PrincipalAccountRequired*

challenge_class_required
Challenge-Klasse erforderlich

Type bool

challenge_structured
Challenge strukturiert

Type bool

initialization_mode
Initialisierungsmodus

Type *fints.formals.InitializationMode*

description_required
Bezeichnung des TAN-Medium erforderlich

Type *fints.formals.DescriptionRequired*

supported_media_number
Anzahl unterstützter aktiver TAN-Medien

Type int

VERSION
TAN mechanism version

1.5 Sending SEPA transfers

1.5.1 Simple mode

You can create a simple SEPA transfer using this convenient client method:

class fints.client.**FinTS3Client**(*bank_identifier*, *user_id*, *customer_id=None*, *from_data: bytes = None*, *product_id=None*, *product_version='4.0.0'*, *mode=<FinTSClientMode.INTERACTIVE: 'interactive'>*)

simple_sepa_transfer(*account: fints.models.SEPAAccount*, *iban: str*, *bic: str*, *recipient_name: str*, *amount: decimal.Decimal*, *account_name: str*, *reason: str*, *instant_payment=False*, *endtoend_id='NOTPROVIDED'*)
Simple SEPA transfer.

Parameters

- **account** – SEPAAccount to start the transfer from.
- **iban** – Recipient's IBAN
- **bic** – Recipient's BIC
- **recipient_name** – Recipient name
- **amount** – Amount as a Decimal
- **account_name** – Sender account name
- **reason** – Transfer reason
- **instant_payment** – Whether to use instant payment (defaults to False)
- **endtoend_id** – End-to-end-Id (defaults to NOTPROVIDED)

Returns Returns either a NeedRetryResponse or TransactionResponse

You should then enter a TAN, read our chapter *Working with TANs* to find out more.

1.5.2 Advanced mode

If you want to use advanced methods, you can supply your own SEPA XML:

class fints.client.**FinTS3Client**(*bank_identifier*, *user_id*, *customer_id=None*, *from_data: bytes = None*, *product_id=None*, *product_version='4.0.0'*, *mode=<FinTSClientMode.INTERACTIVE: 'interactive'>*)

sepa_transfer(*account: fints.models.SEPAAccount*, *pain_message: str*, *multiple=False*, *control_sum=None*, *currency='EUR'*, *book_as_single=False*, *pain_descriptor='urn:iso:std:iso:20022:tech:xsd:pain.001.001.03'*, *instant_payment=False*)

Custom SEPA transfer.

Parameters

- **account** – SEPAAccount to send the transfer from.
- **pain_message** – SEPA PAIN message containing the transfer details.
- **multiple** – Whether this message contains multiple transfers.
- **control_sum** – Sum of all transfers (required if there are multiple)
- **currency** – Transfer currency
- **book_as_single** – Kindly ask the bank to put multiple transactions as separate lines on the bank statement (defaults to `False`)
- **pain_descriptor** – URN of the PAIN message schema used.
- **instant_payment** – Whether this is an instant transfer (defaults to `False`)

Returns Returns either a NeedRetryResponse or TransactionResponse

1.5.3 Full example

```
client = FinTS3PinTanClient(...)
minimal_interactive_cli_bootstrap(client)

with client:
    if client.init_tan_response:
        print("A TAN is required", client.init_tan_response.challenge)

        if getattr(client.init_tan_response, 'challenge_hhduc', None):
            try:
                terminal_flicker_unix(client.init_tan_response.challenge_hhduc)
            except KeyboardInterrupt:
                pass

        tan = input('Please enter TAN:')
        client.send_tan(client.init_tan_response, tan)

    res = client.simple_sepa_transfer(
        account=accounts[0],
        iban='DE12345',
        bic='BIC12345',
        amount=Decimal('7.00'),
        recipient_name='Foo',
        account_name='Test',
```

(continues on next page)

(continued from previous page)

```
        reason='Birthday gift',
        endtoend_id='NOTPROVIDED',
    )

    if isinstance(res, NeedTANResponse):
        print("A TAN is required", res.challenge)

        if getattr(res, 'challenge_hhduc', None):
            try:
                terminal_flicker_unix(res.challenge_hhduc)
            except KeyboardInterrupt:
                pass

        tan = input('Please enter TAN:')
        res = client.send_tan(res, tan)

    print(res.status)
    print(res.responses)
```

1.6 Creating SEPA debits

You can submit a SEPA debit XML file to the bank with the `sepa_debit` method:

class fints.client.**FinTS3Client**(*bank_identifier*, *user_id*, *customer_id=None*, *from_data: bytes = None*, *product_id=None*, *product_version='4.0.0'*, *mode=<FinTSClientMode.INTERACTIVE: 'interactive'>*)

sepa_debit(*account: fints.models.SEPAAccount*, *pain_message: str*, *multiple=False*, *cor1=False*, *control_sum=None*, *currency='EUR'*, *book_as_single=False*, *pain_descriptor='urn:iso:std:iso:20022:tech:xsd:pain.008.003.01'*)

Custom SEPA debit.

Parameters

- **account** – SEPAAccount to send the debit from.
- **pain_message** – SEPA PAIN message containing the debit details.
- **multiple** – Whether this message contains multiple debits.
- **cor1** – Whether to use COR1 debit (lead time reduced to 1 day)
- **control_sum** – Sum of all debits (required if there are multiple)
- **currency** – Debit currency
- **book_as_single** – Kindly ask the bank to put multiple transactions as separate lines on the bank statement (defaults to `False`)
- **pain_descriptor** – URN of the PAIN message schema used. Defaults to `urn:iso:std:iso:20022:tech:xsd:pain.008.003.01`.

Returns Returns either a NeedRetryResponse or TransactionResponse (with data['task_id'] set, if available)

You should then enter a TAN, read our chapter *Working with TANs* to find out more.

1.6.1 Full example

You can easily generate XML using the `sepaxml` python library:

```
from sepaxml import SepaDD

config = {
    "name": "Test Company",
    "IBAN": "DE12345",
    "BIC": "BIC12345",
    "batch": False,
    "creditor_id": "TESTCORPID",
    "currency": "EUR",
}

sepa = SepaDD(config, schema="pain.008.002.02")
sepa.add_payment({
    "name": "Customer",
    "IBAN": "DE12345",
    "BIC": "BIC12345",
    "amount": 100,
    "type": "OOFF",  # FRST, RCUR, OOFF, FNAL
    "collection_date": datetime.date.today() + datetime.timedelta(days=3),
    "mandate_id": "FINTSTEST1",
    "mandate_date": datetime.date(2018, 7, 26),
    "description": "FinTS Test transaction",
})
pain_message = sepa.export().decode()

client = FinTS3PinTanClient(...)
minimal_interactive_cli_bootstrap(client)

with client:
    if client.init_tan_response:
        print("A TAN is required", client.init_tan_response.challenge)

        if getattr(client.init_tan_response, 'challenge_hhduc', None):
            try:
                terminal_flicker_unix(client.init_tan_response.challenge_hhduc)
            except KeyboardInterrupt:
                pass

        tan = input('Please enter TAN:')
        client.send_tan(client.init_tan_response, tan)

    res = client.sepa_debit(
        account=accounts[0],
        data=pain_message,
        multiple=False,
        control_sum=Decimal('1.00'),
        pain_descriptor='urn:iso:std:iso:20022:tech:xsd:pain.008.002.02'
    )

    if isinstance(res, NeedTANResponse):
        print("A TAN is required", res.challenge)

        if getattr(res, 'challenge_hhduc', None):
            try:
```

(continues on next page)

(continued from previous page)

```
                terminal_flicker_unix(res.challenge_hhduc)
            except KeyboardInterrupt:
                pass

        tan = input('Please enter TAN:')
        res = client.send_tan(res, tan)

    print(res.status)
    print(res.responses)
```

1.7 Tested banks

The following banks have been tested with version 3.x of this library:

Bank	Transactions and Balance	Holdings	Transfer	Debits
Postbank	Yes			
BBBank eG	Yes		Yes	
Sparkasse Heidelberg	Yes			
comdirect	Yes			

1.7.1 Tested security functions

- `902` “photoTAN”
- `921` “pushTAN”
- `930` “mobile TAN”
- `942` “mobile TAN”
- `962` “Smart-TAN plus manuell”
- `972` “Smart-TAN plus optisch”

1.7.2 Legacy results

The following banks have been tested with the old version 1.x of this library:

Bank	Statements	Holdings	Transfer	Debits
BBBank eG	Yes		Yes	
CortalConsors	Yes	Yes		
comdirect		Yes		
GLS Bank eG	Yes		Yes	Yes
DKB	Yes			
ING DiBa	Yes			
netbank	Yes			
NIBC Direct	Yes			
Postbank	Yes			
Sparkasse	Yes			
Triodos Bank	Yes			
Volksbank (Fiducia)	Yes			
Wüstenrot	Yes			
1822direkt	Yes	Yes		

The following banks have been tested with the old version 2.x of this library:

Bank	Transactions and Balance	Holdings	Transfer	Debits
GLS Bank eG	Yes		Yes	Yes
Postbank	Yes			
Triodos Bank	Yes		Yes	
Volksbank Darmstadt-Südhessen	Yes		Yes	
Deutsche Skatbank	Yes		Yes	
BBBank eG	Yes		Yes	
MLP Banking AG	Yes			

1.8 Upgrading from python-fints 3.x to 4.x

Release 4.0 of this library was made to introduce a breaking change:

- You now need to register your application with the Deutsche Kreditwirtschaft (German banking association) and supply your assigned product IT when initializing the library.

The library used to have a built-in product ID that was used as a default if you didn't. This was very useful, but Deutsche Kreditwirtschaft asked us to stop doing this, since it undermindes the whole point of the product registration. The ID included in prior versions of the library will be deactivated at some point and stop working.

You cna find more information as well as the registration form on the ZKA Website (only available in German).

1.9 Upgrading from python-fints 2.x to 3.x

Release 3.0 of this library was made to adjust to changes made by the banks as part of their PSD2 implementation in 2019. Here's what you should know when porting your code:

- A TAN can now be required for dialog initialization. In this case, `client.init_tan_response` will contain a `NeedTANResponse`.
- Basically every method of the client class can now return a `NeedTANResponse`, so you should always expect this case and handle it gracefully.

- Since everything can require a TAN, everything requires a standing dialog. Issuing interactive commands outside of a `with client:` statement is now deprecated. It still might work in very few cases, so we didn't disable it, but we do not support it any longer. This affects you mostly when you work with this on a Python REPL or e.g. in a Notebook.

1.10 Upgrading from python-fints 1.x to 2.x

This library has seen a major rewrite in version 2.0 and the API has changed in a lot of places. These are the most important changes to know:

- The `get_statement` method was renamed to `get_transactions`. → *Reading account transactions*
- The `start_simple_sepa_transfer` method was renamed to `simple_sepa_transfer` and no longer takes a TAN method and TAN medium description as an argument. → *Sending SEPA transfers*
- The `start_sepa_transfer` method was renamed to `sepa_transfer` and no longer takes a TAN method and TAN medium description as an argument. The new parameter `pain_descriptor` should be passed with the version of the PAIN format, e.g. `urn:iso:std:iso:20022:tech:xsd:pain.001.001.03`. → *Sending SEPA transfers*
- The `start_sepa_debit` method was renamed to `sepa_debit` and no longer takes a TAN method and TAN medium description as an argument. The new parameter `pain_descriptor` should be passed with the version of the PAIN format, e.g. `urn:iso:std:iso:20022:tech:xsd:pain.008.003.01`. Also, a new parameter `cor1` is optionally available. → *Creating SEPA debits*
- Working with TANs has changed a lot. `get_tan_methos` has been renamed to `get_tan_mechanisms` and has a new return data type. The chosen TAN method is now set on a client level with `set_tan_mechanism` and `set_tan_medium`. You can find more information in the chapter *Working with TANs* and a full example in the chapter *Sending SEPA transfers*.
- Debug logging output now contains parsed syntax structures instead of data blobs and is much easier to read.
- A new parser for FinTS has been added that is more robust and performs more validation.

In exchange, you get a couple of great new features:

- A new method `fints.client.FinTS3Client.get_information()` was added. → *Fetching bank information*
- It is now possible to serialize and store the state of the client to enable multi-step operations in a stateless environment. → *Storing and restoring client state*

1.11 Troubleshooting and bug reporting

The FinTS specification is long and complicated and in many parts leaves things open to interpretation – or sometimes implementors interpret things differently even though they're not really open to interpretation. This is valid for us, but also for the banks. Making the library work with many different banks is hard, and often impossible without access to a test account. Therefore, we ask you for patience when reporting issues with different banks – and you need to be ready that we might not be able to help you because we do not have the time or bank account required to dig deeper.

Therefore, if you run into trouble with this library, you first need to ask yourself a very important question: **Is it me or the library?** To answer this question for most cases, we have attached a script below, that we ask you to use to try the affected feature of the library in a well-documented way. Apart from changing the arguments (i.e. your bank's parameters and your credentials) at the top, we ask you **not to make any modifications**. Pasting this bit by bit into a Jupyter notebook **is a modification**. If your issue does not include information as to whether the script below works or does not work for your bank, **we will close your issue without further comment.**

If the script below does not work for you, there is probably a compatibility issue between this library and your bank. Feel free to open an issue, but make sure the issue title includes the name of the bank and the text includes what operations specifically fail.

If the script below does work for you, there is probably something wrong with your usage of the library or our documentation. Feel free to open an issue, but **include full working example code** that is necessary to reproduce the problem.

Note: Before posting anything on GitHub, make sure it does not contain your username, PIN, IBAN, or similarly sensitive data.

```
import datetime
import getpass
import logging
import sys
from decimal import Decimal

from fints.client import FinTS3PinTanClient, NeedTANResponse,␣
↪FinTSUnsupportedOperation
from fints.hhd.flicker import terminal_flicker_unix
from fints.utils import minimal_interactive_cli_bootstrap

logging.basicConfig(level=logging.DEBUG)

client_args = (
    'REPLACEME',  # BLZ
    'REPLACEME',  # USER
    getpass.getpass('PIN: '),
    'REPLACEME'  # ENDPOINT
)

f = FinTS3PinTanClient(*client_args)
minimal_interactive_cli_bootstrap(f)


def ask_for_tan(response):
    print("A TAN is required")
    print(response.challenge)
    if getattr(response, 'challenge_hhduc', None):
        try:
            terminal_flicker_unix(response.challenge_hhduc)
        except KeyboardInterrupt:
            pass
    tan = input('Please enter TAN:')
    return f.send_tan(response, tan)


# Open the actual dialog
with f:
    # Since PSD2, a TAN might be needed for dialog initialization. Let's check if␣
↪there is one required
    if f.init_tan_response:
        ask_for_tan(f.init_tan_response)

    # Fetch accounts
    accounts = f.get_sepa_accounts()
```

(continues on next page)

(continued from previous page)

```
    if isinstance(accounts, NeedTANResponse):
        accounts = ask_for_tan(accounts)
    if len(accounts) == 1:
        account = accounts[0]
    else:
        print("Multiple accounts available, choose one")
        for i, mm in enumerate(accounts):
            print(i, mm.iban)
        choice = input("Choice: ").strip()
        account = accounts[int(choice)]

    # Test pausing and resuming the dialog
    dialog_data = f.pause_dialog()

client_data = f.deconstruct(including_private=True)

f = FinTS3PinTanClient(*client_args, from_data=client_data)
with f.resume_dialog(dialog_data):
    while True:
        operations = [
            "End dialog",
            "Fetch transactions of the last 30 days",
            "Fetch transactions of the last 120 days",
            "Fetch transactions XML of the last 30 days",
            "Fetch transactions XML of the last 120 days",
            "Fetch information",
            "Fetch balance",
            "Fetch holdings",
            "Fetch scheduled debits",
            "Fetch status protocol",
            "Make a simple transfer"
        ]

        print("Choose an operation")
        for i, o in enumerate(operations):
            print(i, o)
        choice = int(input("Choice: ").strip())
        try:
            if choice == 0:
                break
            elif choice == 1:
                res = f.get_transactions(account, datetime.date.today() - datetime.
↪timedelta(days=30),
                                         datetime.date.today())
                while isinstance(res, NeedTANResponse):
                    res = ask_for_tan(res)
                print("Found", len(res), "transactions")
            elif choice == 2:
                res = f.get_transactions(account, datetime.date.today() - datetime.
↪timedelta(days=120),
                                         datetime.date.today())
                while isinstance(res, NeedTANResponse):
                    res = ask_for_tan(res)
                print("Found", len(res), "transactions")
            elif choice == 3:
                res = f.get_transactions_xml(account, datetime.date.today() -␣
↪datetime.timedelta(days=30),
```

(continues on next page)

(continued from previous page)

```
                                                datetime.date.today())
                while isinstance(res, NeedTANResponse):
                    res = ask_for_tan(res)
                print("Found", len(res[0]) + len(res[1]), "XML documents")
            elif choice == 4:
                res = f.get_transactions_xml(account, datetime.date.today() -␣
↪datetime.timedelta(days=120),
                                                datetime.date.today())
                while isinstance(res, NeedTANResponse):
                    res = ask_for_tan(res)
                print("Found", len(res[0]) + len(res[1]), "XML documents")
            elif choice == 5:
                print(f.get_information())
            elif choice == 6:
                res = f.get_balance(account)
                while isinstance(res, NeedTANResponse):
                    res = ask_for_tan(res)
                print(res)
            elif choice == 7:
                res = f.get_holdings(account)
                while isinstance(res, NeedTANResponse):
                    res = ask_for_tan(res)
                print(res)
            elif choice == 8:
                res = f.get_scheduled_debits(account)
                while isinstance(res, NeedTANResponse):
                    res = ask_for_tan(res)
                print(res)
            elif choice == 9:
                res = f.get_status_protocol()
                while isinstance(res, NeedTANResponse):
                    res = ask_for_tan(res)
                print(res)
            elif choice == 10:
                res = f.simple_sepa_transfer(
                    account=accounts[0],
                    iban=input('Target IBAN:'),
                    bic=input('Target BIC:'),
                    amount=Decimal(input('Amount:')),
                    recipient_name=input('Recipient name:'),
                    account_name=input('Your name:'),
                    reason=input('Reason:'),
                    endtoend_id='NOTPROVIDED',
                )

                if isinstance(res, NeedTANResponse):
                    ask_for_tan(res)
        except FinTSUnsupportedOperation as e:
            print("This operation is not supported by this bank:", e)
```

CHAPTER 2

Library developer documentation content

2.1 Developer documentation/API

This part of the documentation is for you if you want to improve python-fints, but also if you just want to look behind the curtain.

2.1.1 Parsing and serialization

class fints.parser.**FinTS3Parser**
Parser for FinTS/HBCI 3.0 messages

parse_message(*data: bytes*) → fints.types.SegmentSequence
Takes a FinTS 3.0 message as byte array, and returns a parsed segment sequence

class fints.parser.**FinTS3Serializer**
Serializer for FinTS/HBCI 3.0 messages

serialize_message(*message: fints.types.SegmentSequence*) → bytes
Serialize a message (as SegmentSequence, list of FinTS3Segment, or FinTS3Segment) into a byte array

Example usage:

```
>>> message = (b
↪'HNHBK:1:3+000000000428+300+430711670077=043999659571CN9D=+2+430711670077=043'
...             b"999659571CN9D=:2'HNVSK:998:3+PIN:1+998+1+2::oIm3BlHv6mQBAADYgbPpp?
↪+kWrAQA+1+"
...             b"2:2:13:@8@00000000:5:1+280:15050500:hermes:S:0:0+0
↪'HNVSD:999:1+@195@HNSHK:2:"
...             b'4+PIN:1+999+9166926+1+1+2::oIm3BlHv6mQBAADYgbPpp?
↪+kWrAQA+1+1+1:999:1+6:10:16'
...             b"+280:15050500:hermes:S:0:0'HIRMG:3:2+0010::Nachricht␣
↪entgegengenommen.+0100:"
...             b":Dialog beendet.'HNSHA:4:2+9166926''HNHBS:5:1+2'")
>>> from fints.parser import FinTS3Parser
```

(continues on next page)

(continued from previous page)

```
>>> s = FinTS3Parser().parse_message(message)
>>> s
SegmentSequence([fints.segments.HNHBK3(header=fints.formals.SegmentHeader('HNHBK', 1,␣
↪3), message_size='000000000428', hbci_version=300, dialog_id=
↪'430711670077=043999659571CN9D=', message_number=2, reference_message=fints.formals.
↪ReferenceMessage(dialog_id='430711670077=043999659571CN9D=', message_number=2)),␣
↪fints.segments.HNVSK3(header=fints.formals.SegmentHeader('HNVSK', 998, 3), security_
↪profile=fints.formals.SecurityProfile(security_method='PIN', security_method_
↪version=1), security_function='998', security_role='1', security_identification_
↪details=fints.formals.SecurityIdentificationDetails(name_party='2', cid=None,␣
↪identifier_party='oIm3BlHv6mQBAADYgbPpp+kWrAQA'), security_datetime=fints.formals.
↪SecurityDateTime(datetime_type='1'), encryption_algorithm=fints.formals.
↪EncryptionAlgorithm(usage_encryption='2', operation_mode='2', encryption_algorithm=
↪'13', algorithm_parameter_value=b'00000000', algorithm_parameter_name='5',␣
↪algorithm_parameter_iv_name='1'), key_name=fints.formals.KeyName(bank_
↪identifier=fints.formals.BankIdentifier(country_identifier='280', bank_code=
↪'15050500'), user_id='hermes', key_type='S', key_number=0, key_version=0),␣
↪compression_function='0'), fints.segments.HNVSD1(header=fints.formals.SegmentHeader(
↪'HNVSD', 999, 1), data=SegmentSequence([fints.segments.HNSHK4(header=fints.formals.
↪SegmentHeader('HNSHK', 2, 4), security_profile=fints.formals.
↪SecurityProfile(security_method='PIN', security_method_version=1), security_
↪function='999', security_reference='9166926', security_application_area='1',␣
↪security_role='1', security_identification_details=fints.formals.
↪SecurityIdentificationDetails(name_party='2', cid=None, identifier_party=
↪'oIm3BlHv6mQBAADYgbPpp+kWrAQA'), security_reference_number=1, security_
↪datetime=fints.formals.SecurityDateTime(datetime_type='1'), hash_algorithm=fints.
↪formals.HashAlgorithm(usage_hash='1', hash_algorithm='999', algorithm_parameter_
↪name='1'), signature_algorithm=fints.formals.SignatureAlgorithm(usage_signature='6',
↪ signature_algorithm='10', operation_mode='16'), key_name=fints.formals.
↪KeyName(bank_identifier=fints.formals.BankIdentifier(country_identifier='280', bank_
↪code='15050500'), user_id='hermes', key_type='S', key_number=0, key_version=0)),␣
↪fints.segments.HIRMG2(header=fints.formals.SegmentHeader('HIRMG', 3, 2),␣
↪responses=[fints.formals.Response(code='0010', reference_element=None, text=
↪'Nachricht entgegengenommen.'), fints.formals.Response(code='0100', reference_
↪element=None, text='Dialog beendet.')]), fints.segments.HNSHA2(header=fints.formals.
↪SegmentHeader('HNSHA', 4, 2), security_reference='9166926')])), fints.segments.
↪HNHBS1(header=fints.formals.SegmentHeader('HNHBS', 5, 1), message_number=2)])
>>> from fints.parser import FinTS3Serializer
>>> FinTS3Serializer().serialize_message(s)
b
↪"HNHBK:1:3+000000000428+300+430711670077=043999659571CN9D=+2+430711670077=043999659571CN9D=:2
↪'HNVSK:998:3+PIN:1+998+1+2::oIm3BlHv6mQBAADYgbPpp?
↪+kWrAQA+1+2:2:13:@8@00000000:5:1+280:15050500:hermes:S:0:0+0
↪'HNVSD:999:1+@195@HNSHK:2:4+PIN:1+999+9166926+1+1+2::oIm3BlHv6mQBAADYgbPpp?
↪+kWrAQA+1+1+1:999:1+6:10:16+280:15050500:hermes:S:0:0'HIRMG:3:2+0010::Nachricht␣
↪entgegengenommen.+0100::Dialog beendet.'HNSHA:4:2+9166926''HNHBS:5:1+2'"
```

Note: In general parsing followed by serialization is not idempotent: A message may contain empty list elements at the end, but our serializer will never generate them.

2.1.2 FinTS Segments

A segment is the core communication workhorse in FinTS. Each segment has a header of fixed format, which includes the segment type ("Segmentkennung"), number within the message, version, and, optionally, the number of the

segment of another message it is in response or relation to (“Bezugssegment”).

The header is followed by a nested structure of fields and groups of fields, the exact specification of which depends on the segment type and version.

All segment classes derive from *FinTS3Segment*, which specifies the `header` attribute of *SegmentHeader* type.

class fints.segments.base.**FinTS3Segment**(**args*, ***kwargs*)

TYPE
Segment type. Will be determined from the class name in subclasses, if the class name consists only of uppercase characters followed by decimal digits. Subclasses may explicitly set a class attribute instead.

VERSION
Segment version. Will be determined from the class name in subclasses, if the class name consists only of uppercase characters followed by decimal digits. Subclasses may explicitly set a class attribute instead.

classmethod find_subclass(*segment: list*)
Parse the given `segment` parameter as a *SegmentHeader* and return a subclass with matching type and version class attributes.

header
Segmentkopf

Type *fints.formals.SegmentHeader*

print_nested(*stream=None*, *level=0*, *indent=' '*, *prefix=''*, *first_level_indent=True*, *trailer=''*, *print_doc=True*, *first_line_suffix=''*)
Structured nested print of the object to the given stream.

The print-out is eval()able to reconstruct the object.

The *FinTS3Segment* class and its base classes employ a number of dynamic programming techniques so that derived classes need only specify the name, order and type of fields. All type conversion, construction etc. will take place automatically. All derived classes basically should behave “as expected”, returning only native Python datatypes.

Consider this example segment class:

```
class HNHBS1(FinTS3Segment):
    message_number = DataElementField(type='num', max_length=4)
```

Calling `print_nested` on an instance of this class might output:

```
fints.segments.HNHBS1(
    header = fints.formals.SegmentHeader('HNHBS', 4, 1),
    message_number = 1,
)
```

All Segments

fints.segments.accounts module

class fints.segments.accounts.**HISPA1**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

SEPA-Kontoverbindung rückmelden, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HISPA'

VERSION = 1

accounts
SEPA-Kontoverbindung

Type *fints.formals.KTZ1*

class fints.segments.accounts.**HISPAS1**(**args*, ***kwargs*)
Bases: *fints.segments.base.ParameterSegment*

SEPA-Kontoverbindung anfordern, Parameter, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HISPAS'

VERSION = 1

parameter
Parameter SEPA-Kontoverbindung anfordern

Type *fints.formals.GetSEPAAccountParameter1*

class fints.segments.accounts.**HKSPA1**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

SEPA-Kontoverbindung anfordern, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HKSPA'

VERSION = 1

accounts
Kontoverbindung

Type *fints.formals.Account3*

fints.segments.auth module

class fints.segments.auth.**HIPINS1**(**args*, ***kwargs*)
Bases: *fints.segments.base.ParameterSegment*

PIN/TAN-spezifische Informationen, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

TYPE = 'HIPINS'

VERSION = 1

parameter
Parameter PIN/TAN-spezifische Informationen

Type *fints.formals.ParameterPinTan*

class fints.segments.auth.**HITAB4**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

TAN-Generator/Liste anzeigen Bestand Rückmeldung, version 4

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

TYPE = 'HITAB'

VERSION = 4

tan_media_list
TAN-Medium-Liste

Type *fints.formals.TANMedia4*

tan_usage_option
TAN_Einsatzoption

Type *fints.formals.TANUsageOption*

class fints.segments.auth.**HITAB5**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

TAN-Generator/Liste anzeigen Bestand Rückmeldung, version 5

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

TYPE = 'HITAB'

VERSION = 5

tan_media_list
TAN-Medium-Liste

Type *fints.formals.TANMedia5*

tan_usage_option
TAN_Einsatzoption

Type *fints.formals.TANUsageOption*

class fints.segments.auth.**HITAN2**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Zwei-Schritt-TAN-Einreichung Rückmeldung, version 2

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

TYPE = 'HITAN'

VERSION = 2

ben
BEN

Type str

challenge
Challenge

Type str

challenge_valid_until
Gültigkeitsdatum und -uhrzeit für Challenge

Type *fints.formals.ChallengeValidUntil*

tan_list_number
TAN-Listennummer

Type str

tan_process
TAN-Prozess

Type str

task_hash_value
Auftrags-Hashwert

Type bytes

task_reference
Auftragsreferenz

Type str

class fints.segments.auth.**HITAN3**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Zwei-Schritt-TAN-Einreichung Rückmeldung, version 3

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

TYPE = 'HITAN'

VERSION = 3

ben
BEN

Type str

challenge
Challenge

Type str

challenge_valid_until
Gültigkeitsdatum und -uhrzeit für Challenge

Type *fints.formals.ChallengeValidUntil*

tan_list_number
TAN-Listennummer

Type str

tan_medium_name
Bezeichnung des TAN-Mediums

Type str

tan_process
TAN-Prozess

Type str

task_hash_value
Auftrags-Hashwert

Type bytes

task_reference
Auftragsreferenz

Type str

class fints.segments.auth.**HITAN5**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Zwei-Schritt-TAN-Einreichung Rückmeldung, version 5

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

TYPE = 'HITAN'

VERSION = 5

ben
BEN

Type str

challenge
Challenge

Type str

challenge_hhduc
Challenge HHD_UC

Type bytes

challenge_valid_until
Gültigkeitsdatum und -uhrzeit für Challenge

Type *fints.formals.ChallengeValidUntil*

tan_list_number
TAN-Listennummer

Type str

tan_medium_name
Bezeichnung des TAN-Mediums

Type str

tan_process
TAN-Prozess

Type str

task_hash_value
Auftrags-Hashwert

Type bytes

task_reference
Auftragsreferenz

Type str

class fints.segments.auth.**HITAN6**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Zwei-Schritt-TAN-Einreichung Rückmeldung, version 6

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

TYPE = 'HITAN'

VERSION = 6

challenge
Challenge

Type str

challenge_hhduc
Challenge HHD_UC

Type bytes

challenge_valid_until
Gültigkeitsdatum und -uhrzeit für Challenge

Type *fints.formals.ChallengeValidUntil*

tan_medium_name
Bezeichnung des TAN-Mediums

Type str

tan_process
TAN-Prozess

Type str

task_hash_value
Auftrags-Hashwert

Type bytes

task_reference
Auftragsreferenz

Type str

class fints.segments.auth.**HITANS1**(**args*, ***kwargs*)
Bases: *fints.segments.auth.HITANSBase*

TYPE = 'HITANS'

VERSION = 1

parameter

Type *fints.formals.ParameterTwostepTAN1*

class fints.segments.auth.**HITANS2**(**args*, ***kwargs*)
Bases: *fints.segments.auth.HITANSBase*

TYPE = 'HITANS'

VERSION = 2

parameter

Type *fints.formals.ParameterTwostepTAN2*

class fints.segments.auth.**HITANS3**(**args*, ***kwargs*)
Bases: *fints.segments.auth.HITANSBase*

TYPE = 'HITANS'

VERSION = 3

parameter

Type *fints.formals.ParameterTwostepTAN3*

class fints.segments.auth.**HITANS4**(**args*, ***kwargs*)
Bases: *fints.segments.auth.HITANSBase*

TYPE = 'HITANS'

VERSION = 4

parameter

Type *fints.formals.ParameterTwostepTAN4*

class fints.segments.auth.**HITANS5**(**args*, ***kwargs*)
Bases: *fints.segments.auth.HITANSBase*

TYPE = 'HITANS'

VERSION = 5

parameter

Type *fints.formals.ParameterTwostepTAN5*

class fints.segments.auth.**HITANS6**(**args*, ***kwargs*)
Bases: *fints.segments.auth.HITANSBase*

TYPE = 'HITANS'

VERSION = 6

parameter

Type *fints.formals.ParameterTwostepTAN6*

class fints.segments.auth.**HITANSBase**(**args*, ***kwargs*)
Bases: *fints.segments.base.ParameterSegment*

TYPE = None

VERSION = None

class fints.segments.auth.**HKIDN2**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Identifikation, version 2

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Formals

TYPE = 'HKIDN'

VERSION = 2

bank_identifier
Kreditinstitutskennung

Type *fints.formals.BankIdentifier*

customer_id
Kunden-ID

Type str

system_id
Kundensystem-ID

Type str

system_id_status
Kundensystem-Status

Type *fints.formals.SystemIDStatus*

class fints.segments.auth.**HKTAB4**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

TAN-Generator/Liste anzeigen Bestand, version 4

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

TYPE = 'HKTAB'

VERSION = 4

tan_media_class
TAN-Medium-Klasse

Type *fints.formals.TANMediaClass3*

tan_media_type
TAN-Medium-Art

Type *fints.formals.TANMediaType2*

class fints.segments.auth.**HKTAB5**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

TAN-Generator/Liste anzeigen Bestand, version 5

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

TYPE = 'HKTAB'

VERSION = 5

tan_media_class
TAN-Medium-Klasse

Type *fints.formals.TANMediaClass4*

tan_media_type
TAN-Medium-Art

Type *fints.formals.TANMediaType2*

class fints.segments.auth.**HKTAN2**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Zwei-Schritt-TAN-Einreichung, version 2

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

TYPE = 'HKTAN'

VERSION = 2

cancel_task
Auftrag stornieren

Type bool

challenge_class
Challenge-Klasse

Type int

further_tan_follows
Weitere TAN folgt

Type bool

parameter_challenge_class
Parameter Challenge-Klasse

Type *fints.formals.ParameterChallengeClass*

tan_list_number
TAN-Listennummer

Type str

tan_process
TAN-Prozess

Type str

task_hash_value
Auftrags-Hashwert

Type bytes

task_reference
Auftragsreferenz

Type str

class fints.segments.auth.**HKTAN3**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Zwei-Schritt-TAN-Einreichung, version 3

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

TYPE = 'HKTAN'

VERSION = 3

cancel_task
Auftrag stornieren

Type bool

challenge_class
Challenge-Klasse

Type int

further_tan_follows
Weitere TAN folgt

Type bool

parameter_challenge_class
Parameter Challenge-Klasse

Type *fints.formals.ParameterChallengeClass*

tan_list_number
TAN-Listennummer

Type str

tan_medium_name
Bezeichnung des TAN-Mediums

Type str

tan_process
TAN-Prozess

Type str

task_hash_value
Auftrags-Hashwert

Type bytes

task_reference
Auftragsreferenz

Type str

class fints.segments.auth.**HKTAN5**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Zwei-Schritt-TAN-Einreichung, version 5

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

TYPE = 'HKTAN'

VERSION = 5

account
Kontoverbindung international Auftraggeber

Type *fints.formals.KTI1*

cancel_task
Auftrag stornieren

Type bool

challenge_class
Challenge-Klasse

Type int

further_tan_follows
Weitere TAN folgt

Type bool

parameter_challenge_class
Parameter Challenge-Klasse

Type *fints.formals.ParameterChallengeClass*

segment_type
Segmentkennung

Type str

sms_charge_account
SMS-Abbuchungskonto

Type *fints.formals.KTI1*

tan_list_number
TAN-Listennummer

Type str

tan_medium_name
Bezeichnung des TAN-Mediums

Type str

tan_process
TAN-Prozess

Type str

task_hash_value
Auftrags-Hashwert

Type bytes

task_reference
Auftragsreferenz

Type str

class fints.segments.auth.**HKTAN6**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Zwei-Schritt-TAN-Einreichung, version 6

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

TYPE = 'HKTAN'

VERSION = 6

account
Kontoverbindung international Auftraggeber

Type *fints.formals.KTI1*

cancel_task
Auftrag stornieren

Type bool

challenge_class
Challenge-Klasse

Type int

further_tan_follows
Weitere TAN folgt

Type bool

parameter_challenge_class
Parameter Challenge-Klasse

Type *fints.formals.ParameterChallengeClass*

response_hhd_uc
Antwort HHD_UC

Type *fints.formals.ResponseHHDUC*

segment_type
Segmentkennung

Type str

sms_charge_account
SMS-Abbuchungskonto

Type *fints.formals.KTI1*

tan_medium_name
Bezeichnung des TAN-Mediums

Type str

tan_process
TAN-Prozess

Type str

task_hash_value
Auftrags-Hashwert

Type bytes

task_reference
Auftragsreferenz

Type str

class fints.segments.auth.**HKVVB3**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Verarbeitungsvorbereitung, version 3

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Formals

TYPE = 'HKVVB'

VERSION = 3

bpd_version
BPD-Version

Type int

language
Dialogsprache

Type *fints.formals.Language2*

product_name
Produktbezeichnung

Type str

product_version
Produktversion

Type str

upd_version
UPD-Version

Type int

fints.segments.bank module

class fints.segments.bank.**HIBPA3**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Bankparameter allgemein, version 3

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Formals

TYPE = 'HIBPA'

VERSION = 3

bank_identifier
Kreditinstitutskennung

Type *fints.formals.BankIdentifier*

bank_name
Kreditinstitutsbezeichnung

Type str

bpd_version
BPD-Version

Type int

max_message_length
Maximale Nachrichtengröße

Type int

max_timeout
Maximaler Timeout-Wert

Type int

min_timeout
Minimaler Timeout-Wert

Type int

number_tasks
Anzahl Geschäftsvorfallarten pro Nachricht

Type int

supported_hbci_version
Unterstützte HBCI-Versionen

Type *fints.formals.SupportedHBCIVersions2*

supported_languages
Unterstützte Sprachen

Type *fints.formals.SupportedLanguages2*

class fints.segments.bank.**HIKOM4**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Kommunikationszugang rückmelden, version 4

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Formals

TYPE = 'HIKOM'

VERSION = 4

bank_identifier
Kreditinstitutskennung

Type *fints.formals.BankIdentifier*

communication_parameters
Kommunikationsparameter

Type *fints.formals.CommunicationParameter2*

default_language
Standardsprache

Type *fints.formals.Language2*

class fints.segments.bank.**HIUPA4**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Userparameter allgemein

TYPE = 'HIUPA'

VERSION = 4

extension
Erweiterung, allgemein

Type str

upd_usage
UPD-Verwendung

Type *fints.formals.UPDUsage*

upd_version
UPD-Version

Type int

user_identifier
Benutzerkennung

Type str

username
Benutzername

Type str

class fints.segments.bank.**HIUPD6**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Kontoinformationen

TYPE = 'HIUPD'

VERSION = 6

account_currency
Kontowährung

Type str

account_information
Kontoverbindung

Type *fints.formals.AccountInformation*

account_limit
Kontolimit

Type *fints.formals.AccountLimit*

account_product_name
Kontoproduktbezeichnung

Type str

account_type
Kontoart

Type int

allowed_transactions
Erlaubte Geschäftsvorfälle

Type *fints.formals.AllowedTransaction*

customer_id
Kunden-ID

Type str

extension
Erweiterung, kontobezogen

Type str

iban
IBAN

Type str

name_account_owner_1
Name des Kontoinhabers 1

Type str

name_account_owner_2
Name des Kontoinhabers 2

Type str

class fints.segments.bank.**HKKOM4**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Kommunikationszugang anfordern, version 4

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Formals

TYPE = 'HKKOM'

VERSION = 4

end_bank_identifier
Bis Kreditinstitutskennung

Type *fints.formals.BankIdentifier*

max_number_responses
Maximale Anzahl Einträge

Type int

start_bank_identifier
Von Kreditinstitutskennung

Type *fints.formals.BankIdentifier*

touchdown_point
Aufsetzpunkt

Type str

fints.segments.base module

class fints.segments.base.**FinTS3SegmentMeta**
Bases: *fints.types.ContainerMeta*

mro()
Return a type's method resolution order.

class fints.segments.base.**ParameterSegment**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

TYPE = None

VERSION = None

max_number_tasks
Maximale Anzahl Aufträge

Type int

min_number_signatures
Anzahl Signaturen mindestens

Type int

security_class
Sicherheitsklasse

Type *fints.formals.SecurityClass*

class fints.segments.base.**ParameterSegment_22**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

TYPE = None

VERSION = None

max_number_tasks
Maximale Anzahl Aufträge

Type int

min_number_signatures
Anzahl Signaturen mindestens

Type int

fints.segments.debit module

class fints.segments.debit.**BatchDebitBase**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

TYPE = None

VERSION = None

account
Kontoverbindung international

Type *fints.formals.KTI1*

request_single_booking
Einzelbuchung gewünscht

Type bool

sepa_descriptor
SEPA Descriptor

Type str

sepa_pain_message
SEPA pain message

Type bytes

sum_amount
Summenfeld

Type *fints.formals.Amount1*

class fints.segments.debit.**DebitResponseBase**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

TYPE = None

VERSION = None

task_id
Auftragsidentifikation

Type str

class fints.segments.debit.**HIDBS1**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Bestand terminierter SEPA-Einzellastschriften rückmelden, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HIDBS'

VERSION = 1

account
Kontoverbindung international

Type *fints.formals.KTI1*

sepa_descriptor
SEPA Descriptor

Type str

sepa_pain_message
SEPA pain message

Type bytes

task_cancelable
Auftrag löschbar

Type bool

task_changeable
Auftrag änderbar

Type bool

task_id
Auftragsidentifikation

Type str

class fints.segments.debit.**HIDBS2**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Bestand terminierter SEPA-Einzellastschriften rückmelden, version 2

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HIDBS'

VERSION = 2

account
Kontoverbindung international

Type *fints.formals.KTI1*

sepa_c_code
SEPA-C-Code

Type *fints.formals.SEPACCode1*

sepa_descriptor
SEPA Descriptor

Type str

sepa_pain_message
SEPA pain message

Type bytes

status_sepa_task
Status SEPA-Auftrag

Type *fints.formals.StatusSEPATask1*

task_changeable
Auftrag änderbar

Type bool

task_id
Auftragsidentifikation

Type str

class fints.segments.debit.**HIDBSS1**(**args*, ***kwargs*)
Bases: *fints.segments.base.ParameterSegment*

Bestand terminierter SEPA-Einzellastschriften Parameter, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HIDBSS'

VERSION = 1

parameter
Parameter Bestand terminierter SEPA-Einzellastschriften

Type *fints.formals.QueryScheduledDebitParameter1*

class fints.segments.debit.**HIDBSS2**(**args*, ***kwargs*)
Bases: *fints.segments.base.ParameterSegment*

Bestand terminierter SEPA-Einzellastschriften Parameter, version 2

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HIDBSS'

VERSION = 2

parameter
Parameter Bestand terminierter SEPA-Einzellastschriften

Type *fints.formals.QueryScheduledDebitParameter2*

class fints.segments.debit.**HIDMB1**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Bestand terminierter SEPA-Sammellastschriften rückmelden, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HIDMB'

VERSION = 1

account
Kontoverbindung international

Type *fints.formals.KTI1*

date_booked
Ausführungsdatum

Type datetime.date

date_entered
Einreichungsdatum

Type datetime.date

debit_count
Anzahl der Aufträge

Type int

sum_amount
Summe der Beträge

Type *fints.formals.Amount1*

task_id
Auftragsidentifikation

Type str

class fints.segments.debit.**HIDMBS1**(**args*, ***kwargs*)
Bases: *fints.segments.base.ParameterSegment*

Bestand terminierter SEPA-Sammellastschriften Parameter, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HIDMBS'

VERSION = 1

parameter
Parameter Bestand terminierter SEPA-Sammellastschriften

Type *fints.formals.QueryScheduledBatchDebitParameter1*

class fints.segments.debit.**HIDMC1**(**args*, ***kwargs*)
Bases: *fints.segments.debit.DebitResponseBase*

Einreichung terminierter SEPA-COR1-Sammellastschrift bestätigen, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HIDMC'

VERSION = 1

class fints.segments.debit.**HIDMCS1**(**args*, ***kwargs*)
Bases: *fints.segments.base.ParameterSegment*

Terminierte SEPA-COR1-Sammellastschrift Parameter, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HIDMCS'

VERSION = 1

parameter
Parameter terminierte SEPA-COR1-Sammellastschrift

Type *fints.formals.ScheduledCOR1BatchDebitParameter1*

class fints.segments.debit.**HIDME1**(**args*, ***kwargs*)
Bases: *fints.segments.debit.DebitResponseBase*

Einreichung terminierter SEPA-Sammellastschrift bestätigen, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HIDME'

VERSION = 1

class fints.segments.debit.**HIDME2**(**args*, ***kwargs*)
Bases: *fints.segments.debit.DebitResponseBase*

Einreichung terminierter SEPA-Sammellastschrift bestätigen, version 2

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HIDME'

VERSION = 2

class fints.segments.debit.**HIDMES1**(**args*, ***kwargs*)
Bases: *fints.segments.base.ParameterSegment*

Terminierte SEPA-Sammellastschrift einreichen Parameter, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HIDMES'

VERSION = 1

parameter
Parameter terminierte SEPA-Sammellastschrift einreichen

Type *fints.formals.ScheduledBatchDebitParameter1*

class fints.segments.debit.**HIDMES2**(**args*, ***kwargs*)
Bases: *fints.segments.base.ParameterSegment*

Terminierte SEPA-Sammellastschrift einreichen Parameter, version 2

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HIDMES'

VERSION = 2

parameter
Parameter terminierte SEPA-Sammellastschrift einreichen

Type *fints.formals.ScheduledBatchDebitParameter2*

class fints.segments.debit.**HIDSC1**(**args*, ***kwargs*)
Bases: *fints.segments.debit.DebitResponseBase*

Einreichung terminierter SEPA-COR1-Einzellastschrift bestätigen, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HIDSC'

VERSION = 1

class fints.segments.debit.**HIDSCS1**(**args*, ***kwargs*)
Bases: *fints.segments.base.ParameterSegment*

Terminierte SEPA-COR1-Einzellastschrift Parameter, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HIDSCS'

VERSION = 1

parameter
Parameter terminierte SEPA-COR1-Einzellastschrift

Type *fints.formals.ScheduledCOR1DebitParameter1*

class fints.segments.debit.**HIDSE1**(**args*, ***kwargs*)
Bases: *fints.segments.debit.DebitResponseBase*

Einreichung terminierter SEPA-Einzellastschrift bestätigen, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HIDSE'

VERSION = 1

class fints.segments.debit.**HIDSE2**(**args*, ***kwargs*)
Bases: *fints.segments.debit.DebitResponseBase*

Einreichung terminierter SEPA-Einzellastschrift bestätigen, version 2

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HIDSE'

VERSION = 2

class fints.segments.debit.**HIDSES1**(**args*, ***kwargs*)
Bases: *fints.segments.base.ParameterSegment*

Terminierte SEPA-Einzellastschrift einreichen Parameter, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HIDSES'

VERSION = 1

parameter
Parameter terminierte SEPA-Sammellastschrift einreichen

Type *fints.formals.ScheduledDebitParameter1*

class fints.segments.debit.**HIDSES2**(**args*, ***kwargs*)
Bases: *fints.segments.base.ParameterSegment*

Terminierte SEPA-Einzellastschrift einreichen Parameter, version 2

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HIDSES'

VERSION = 2

parameter
Parameter terminierte SEPA-Sammellastschrift einreichen

Type *fints.formals.ScheduledDebitParameter2*

class fints.segments.debit.**HKDBS1**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Bestand terminierter SEPA-Einzellastschriften anfordern, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HKDBS'

VERSION = 1

account
Kontoverbindung international

Type *fints.formals.KTI1*

date_end
Bis Datum

Type datetime.date

date_start
Von Datum

Type datetime.date

max_number_responses
Maximale Anzahl Einträge

Type int

supported_sepa_pain_messages
Unterstützte SEPA pain messages

Type *fints.formals.SupportedSEPAPainMessages1*

touchdown_point
Aufsetzpunkt

Type str

class fints.segments.debit.**HKDBS2**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Bestand terminierter SEPA-Einzellastschriften anfordern, version 2

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HKDBS'

VERSION = 2

account
Kontoverbindung international

Type *fints.formals.KTI1*

date_end
Bis Datum

Type datetime.date

date_start
Von Datum

Type datetime.date

max_number_responses
Maximale Anzahl Einträge

Type int

supported_sepa_pain_messages
Unterstützte SEPA pain messages

Type *fints.formals.SupportedSEPAPainMessages1*

touchdown_point
Aufsetzpunkt

Type str

class fints.segments.debit.**HKDMB1**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Bestand terminierter SEPA-Sammellastschriften anfordern, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HKDMB'

VERSION = 1

account
Kontoverbindung international

Type *fints.formals.KTI1*

date_end
Bis Datum

Type datetime.date

date_start
Von Datum

Type datetime.date

max_number_responses
Maximale Anzahl Einträge

Type int

touchdown_point
Aufsetzpunkt

Type str

class fints.segments.debit.**HKDMC1**(**args*, ***kwargs*)
Bases: *fints.segments.debit.BatchDebitBase*

Terminierte SEPA-COR1-Sammellastschrift einreichen, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HKDMC'

VERSION = 1

class fints.segments.debit.**HKDME1**(**args*, ***kwargs*)
Bases: *fints.segments.debit.BatchDebitBase*

Einreichung terminierter SEPA-Sammellastschrift, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HKDME'

VERSION = 1

class fints.segments.debit.**HKDME2**(**args*, ***kwargs*)
Bases: *fints.segments.debit.BatchDebitBase*

Einreichung terminierter SEPA-Sammellastschrift, version 2

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HKDME'

VERSION = 2

class fints.segments.debit.**HKDSC1**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Terminierte SEPA-COR1-Einzellastschrift einreichen, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HKDSC'

VERSION = 1

account
Kontoverbindung international

Type *fints.formals.KTI1*

sepa_descriptor
SEPA Descriptor

Type str

sepa_pain_message
SEPA pain message

Type bytes

class fints.segments.debit.**HKDSE1**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Terminierte SEPA-Einzellastschrift einreichen, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HKDSE'

VERSION = 1

account
Kontoverbindung international

Type *fints.formals.KTI1*

sepa_descriptor
SEPA Descriptor

Type str

sepa_pain_message
SEPA pain message

Type bytes

class fints.segments.debit.**HKDSE2**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Terminierte SEPA-Einzellastschrift einreichen, version 2

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HKDSE'

VERSION = 2

account
Kontoverbindung international

Type *fints.formals.KTI1*

sepa_descriptor
SEPA Descriptor

Type str

sepa_pain_message
SEPA pain message

Type bytes

fints.segments.depot module

class fints.segments.depot.**HIWPD5**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Depotaufstellung rückmelden, version 5

Source: HBCI Homebanking-Computer-Interface, Schnittstellenspezifikation

TYPE = 'HIWPD'

VERSION = 5

holdings
Depotaufstellung

Type bytes

class fints.segments.depot.**HIWPD6**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Depotaufstellung rückmelden, version 6

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HIWPD'

VERSION = 6

holdings
Depotaufstellung

Type bytes

class fints.segments.depot.**HKWPD5**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Depotaufstellung anfordern, version 5

Source: HBCI Homebanking-Computer-Interface, Schnittstellenspezifikation

TYPE = 'HKWPD'

VERSION = 5

account
Depot

Type *fints.formals.Account2*

currency
Währung der Depotaufstellung

Type str

max_number_responses
Maximale Anzahl Einträge

Type int

quality
Kursqualität

Type int

touchdown_point
Aufsetzpunkt

Type str

class fints.segments.depot.**HKWPD6**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Depotaufstellung anfordern, version 6

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HKWPD'

VERSION = 6

account
Depot

Type *fints.formals.Account3*

currency
Währung der Depotaufstellung

Type str

max_number_responses
Maximale Anzahl Einträge

Type int

quality
Kursqualität

Type str

touchdown_point
Aufsetzpunkt

Type str

fints.segments.dialog module

class fints.segments.dialog.**HIRMG2**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Rückmeldungen zur Gesamtnachricht

TYPE = 'HIRMG'

VERSION = 2

responses
Rückmeldung

Type *fints.formals.Response*

class fints.segments.dialog.**HIRMS2**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Rückmeldungen zu Segmenten

TYPE = 'HIRMS'

VERSION = 2

responses
Rückmeldung

Type *fints.formals.Response*

class fints.segments.dialog.**HISYN4**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Synchronisierungsantwort

TYPE = 'HISYN'

VERSION = 4

message_number
Nachrichtennummer

Type int

security_reference_digital_signature
Sicherheitsreferenznummer für Digitale Signatur

Type int

security_reference_signature_key
Sicherheitsreferenznummer für Signierschlüssel

Type int

system_id
Kundensystem-ID

Type str

class fints.segments.dialog.**HKEND1**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Dialogende, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Formals

TYPE = 'HKEND'

VERSION = 1

dialog_id
Dialog-ID

Type str

class fints.segments.dialog.**HKSYN3**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Synchronisierung, version 3

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Formals

TYPE = 'HKSYN'

VERSION = 3

synchronization_mode

Type *fints.formals.SynchronizationMode*

fints.segments.journal module

class fints.segments.journal.**HIPRO3**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Statusprotokoll rückmelden, version 3

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Formals

TYPE = 'HIPRO'

VERSION = 3

date
Datum

Type datetime.date

reference
Bezugssegment

Type int

reference_message
Bezugsnachricht

Type *fints.formals.ReferenceMessage*

responses
Rückmeldung

Type *fints.formals.Response*

time
Uhrzeit

Type datetime.time

class fints.segments.journal.**HIPRO4**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Statusprotokoll rückmelden, version 4

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Formals

TYPE = 'HIPRO'

VERSION = 4

date
Datum

Type datetime.date

reference
Bezugssegment

Type int

reference_message
Bezugsnachricht

Type *fints.formals.ReferenceMessage*

responses
Rückmeldung

Type *fints.formals.Response*

time
Uhrzeit

Type datetime.time

class fints.segments.journal.**HIPROS3**(**args*, ***kwargs*)
Bases: *fints.segments.base.ParameterSegment_22*

Statusprotokoll Parameter, version 3

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Formals

TYPE = 'HIPROS'

VERSION = 3

class fints.segments.journal.**HIPROS4**(**args*, ***kwargs*)
Bases: *fints.segments.base.ParameterSegment*

Statusprotokoll Parameter, version 4

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Formals

TYPE = 'HIPROS'

VERSION = 4

class fints.segments.journal.**HKPRO3**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Statusprotokoll anfordern, version 3

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Formals

TYPE = 'HKPRO'

VERSION = 3

date_end
Bis Datum

Type datetime.date

date_start
Von Datum

Type datetime.date

max_number_responses
Maximale Anzahl Einträge

Type int

touchdown_point
Aufsetzpunkt

Type str

class fints.segments.journal.**HKPRO4**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Statusprotokoll anfordern, version 4

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Formals

TYPE = 'HKPRO'

VERSION = 4

date_end
Bis Datum

Type datetime.date

date_start
Von Datum

Type datetime.date

max_number_responses
Maximale Anzahl Einträge

Type int

touchdown_point
Aufsetzpunkt

Type str

fints.segments.message module

class fints.segments.message.**HNHBK3**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Nachrichtenkopf

TYPE = 'HNHBK'

VERSION = 3

dialog_id
Dialog-ID

Type str

hbci_version
HBCI-Version

Type int

message_number
Nachrichtennummer

Type int

message_size
Größe der Nachricht (nach Verschlüsselung und Komprimierung)

Type int

reference_message
Bezugsnachricht

Type *fints.formals.ReferenceMessage*

class fints.segments.message.**HNHBS1**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Nachrichtenabschluss

TYPE = 'HNHBS'

VERSION = 1

message_number
Nachrichtennummer

Type int

class fints.segments.message.**HNSHA2**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Signaturabschluss, version 2

Source: FinTS Financial Transaction Services, Sicherheitsverfahren HBCI

TYPE = 'HNSHA'

VERSION = 2

security_reference
Sicherheitskontrollreferenz

Type str

user_defined_signature
Benutzerdefinierte Signatur

Type *fints.formals.UserDefinedSignature*

validation_result
Validierungsresultat

Type bytes

class fints.segments.message.**HNSHK4**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Signaturkopf, version 4

Source: FinTS Financial Transaction Services, Sicherheitsverfahren HBCI

TYPE = 'HNSHK'

VERSION = 4

certificate
Zertifikat

Type *fints.formals.Certificate*

hash_algorithm
Hashalgorithmus

Type *fints.formals.HashAlgorithm*

key_name
Schlüsselname

Type *fints.formals.KeyName*

security_application_area
Bereich der Sicherheitsapplikation, kodiert

Type *fints.formals.SecurityApplicationArea*

security_datetime
Sicherheitsdatum und -uhrzeit

Type *fints.formals.SecurityDateTime*

security_function
Sicherheitsfunktion, kodiert

Type str

security_identification_details
Sicherheitsidentifikation, Details

Type *fints.formals.SecurityIdentificationDetails*

security_profile
Sicherheitsprofil

Type *fints.formals.SecurityProfile*

security_reference
Sicherheitskontrollreferenz

Type str

security_reference_number
Sicherheitsreferenznummer

Type int

security_role
Rolle des Sicherheitslieferanten, kodiert

Type *fints.formals.SecurityRole*

signature_algorithm
Signaturalgorithmus

Type *fints.formals.SignatureAlgorithm*

class fints.segments.message.**HNVSD1**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Verschlüsselte Daten, version 1

Source: FinTS Financial Transaction Services, Sicherheitsverfahren HBCI

TYPE = 'HNVSD'

VERSION = 1

data
Daten, verschlüsselt

class fints.segments.message.**HNVSK3**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Verschlüsselungskopf, version 3

Source: FinTS Financial Transaction Services, Sicherheitsverfahren HBCI

TYPE = 'HNVSK'

VERSION = 3

certificate
Zertifikat

Type *fints.formals.Certificate*

compression_function
Komprimierungsfunktion

Type *fints.formals.CompressionFunction*

encryption_algorithm
Verschlüsselungsalgorithmus

Type *fints.formals.EncryptionAlgorithm*

key_name
Schlüsselname

Type *fints.formals.KeyName*

security_datetime
Sicherheitsdatum und -uhrzeit

Type *fints.formals.SecurityDateTime*

security_function
Sicherheitsfunktion, kodiert

Type str

security_identification_details
Sicherheitsidentifikation, Details

Type *fints.formals.SecurityIdentificationDetails*

security_profile
Sicherheitsprofil

Type *fints.formals.SecurityProfile*

security_role
Rolle des Sicherheitslieferanten, kodiert

Type *fints.formals.SecurityRole*

fints.segments.saldo module

class fints.segments.saldo.**HISAL5**(**args*, ***kwargs*)

Bases: *fints.segments.base.FinTS3Segment*

Saldenrückmeldung, version 5

Source: HBCI Homebanking-Computer-Interface, Schnittstellenspezifikation

TYPE = 'HISAL'

VERSION = 5

account
Kontoverbindung Auftraggeber

Type *fints.formals.Account2*

account_product
Kontoproduktbezeichnung

Type str

available_amount
Verfügbarer Betrag

Type *fints.formals.Amount1*

balance_booked
Gebuchter Saldo

Type *fints.formals.Balance1*

balance_pending
Saldo der vorgemerkten Umsätze

Type *fints.formals.Balance1*

booking_date
Buchungsdatum des Saldos

Type datetime.date

booking_time
Buchungsuhrzeit des Saldos

Type datetime.time

currency
Kontowährung

Type str

date_due
Fälligkeit

Type datetime.date

line_of_credit
Kreditlinie

Type *fints.formals.Amount1*

used_amount
Bereits verfügter Betrag

Type *fints.formals.Amount1*

class fints.segments.saldo.**HISAL6**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Saldenrückmeldung, version 6

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HISAL'

VERSION = 6

account
Kontoverbindung Auftraggeber

Type *fints.formals.Account3*

account_product
Kontoproduktbezeichnung

Type str

available_amount
Verfügbarer Betrag

Type *fints.formals.Amount1*

balance_booked
Gebuchter Saldo

Type *fints.formals.Balance2*

balance_pending
Saldo der vorgemerkten Umsätze

Type *fints.formals.Balance2*

booking_timestamp
Buchungszeitpunkt

Type *fints.formals.Timestamp1*

currency
Kontowährung

Type str

date_due
Fälligkeit

Type datetime.date

line_of_credit
Kreditlinie

Type *fints.formals.Amount1*

overdraft
Überziehung

Type *fints.formals.Amount1*

used_amount
Bereits verfügter Betrag

Type *fints.formals.Amount1*

class fints.segments.saldo.**HISAL7**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Saldenrückmeldung, version 7

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HISAL'

VERSION = 7

account
Kontoverbindung international

Type *fints.formals.KTI1*

account_product
Kontoproduktbezeichnung

Type str

available_amount
Verfügbarer Betrag

Type *fints.formals.Amount1*

balance_booked
Gebuchter Saldo

Type *fints.formals.Balance2*

balance_pending
Saldo der vorgemerkten Umsätze

Type *fints.formals.Balance2*

booking_timestamp
Buchungszeitpunkt

Type *fints.formals.Timestamp1*

currency
Kontowährung

Type str

date_due
Fälligkeit

Type datetime.date

line_of_credit
Kreditlinie

Type *fints.formals.Amount1*

overdraft
Überziehung

Type *fints.formals.Amount1*

used_amount
Bereits verfügter Betrag

Type *fints.formals.Amount1*

class fints.segments.saldo.**HKSAL5**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Saldenabfrage, version 5

Source: HBCI Homebanking-Computer-Interface, Schnittstellenspezifikation

TYPE = 'HKSAL'

VERSION = 5

account
Kontoverbindung Auftraggeber

Type *fints.formals.Account2*

all_accounts
Alle Konten

Type bool

max_number_responses
Maximale Anzahl Einträge

Type int

touchdown_point
Aufsetzpunkt

Type str

class fints.segments.saldo.**HKSAL6**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Saldenabfrage, version 6

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HKSAL'

VERSION = 6

account
Kontoverbindung Auftraggeber

Type *fints.formals.Account3*

all_accounts
Alle Konten

Type bool

max_number_responses
Maximale Anzahl Einträge

Type int

touchdown_point
Aufsetzpunkt

Type str

class fints.segments.saldo.**HKSAL7**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Saldenabfrage, version 7

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HKSAL'

VERSION = 7

account
Kontoverbindung international

Type *fints.formals.KTI1*

all_accounts
Alle Konten

Type bool

max_number_responses
Maximale Anzahl Einträge

Type int

touchdown_point
Aufsetzpunkt

Type str

fints.segments.statement module

class fints.segments.statement.**DIKKU2**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Kreditkartenumsätze rückmelden, version 2

Source: Reverse engineered

TYPE = 'DIKKU'

VERSION = 2

class fints.segments.statement.**DIKKUS2**(**args*, ***kwargs*)
Bases: *fints.segments.base.ParameterSegment*

Kreditkartenumsätze anfordern Parameter, version 2

Source: Reverse engineered

TYPE = 'DIKKUS'

VERSION = 2

parameter
Parameter Kreditkartenumsätze anfordern

Type *fints.formals.QueryCreditCardStatements2*

class fints.segments.statement.**DKKKU2**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Kreditkartenumsätze anfordern, version 2

Source: Reverse engineered

TYPE = 'DKKKU'

VERSION = 2

account
Kontoverbindung Auftraggeber

Type *fints.formals.Account2*

credit_card_number
Kreditkartennummer

Type str

date_end
Bis Datum

Type datetime.date

date_start
Von Datum

Type datetime.date

max_number_responses
Maximale Anzahl Einträge

Type int

subaccount
Subaccount?

Type str

touchdown_point
Aufsetzpunkt

Type str

class fints.segments.statement.**HICAZ1**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Kontoumsätze rückmelden/Zeitraum, version 1

Source: HBCI Homebanking-Computer-Interface, Schnittstellenspezifikation

TYPE = 'HICAZ'

VERSION = 1

account
Kontoverbindung Auftraggeber

Type *fints.formals.KTI1*

camt_descriptor
camt-Deskriptor

Type str

statement_booked
Gebuchte Umsätze

Type *fints.formals.BookedCamtStatements1*

statement_pending
Nicht gebuchte Umsätze

Type bytes

class fints.segments.statement.**HIKAZ5**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Kontoumsätze rückmelden/Zeitraum, version 5

Source: HBCI Homebanking-Computer-Interface, Schnittstellenspezifikation

TYPE = 'HIKAZ'

VERSION = 5

statement_booked
Gebuchte Umsätze

Type bytes

statement_pending
Nicht gebuchte Umsätze

Type bytes

class fints.segments.statement.**HIKAZ6**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Kontoumsätze rückmelden/Zeitraum, version 6

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HIKAZ'

VERSION = 6

statement_booked
Gebuchte Umsätze

Type bytes

statement_pending
Nicht gebuchte Umsätze

Type bytes

class fints.segments.statement.**HIKAZ7**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Kontoumsätze rückmelden/Zeitraum, version 7

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HIKAZ'

VERSION = 7

statement_booked
Gebuchte Umsätze

Type bytes

statement_pending
Nicht gebuchte Umsätze

Type bytes

class fints.segments.statement.**HKCAZ1**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Kontoumsätze anfordern/Zeitraum, version 5

Source: HBCI Homebanking-Computer-Interface, Schnittstellenspezifikation

TYPE = 'HKCAZ'

VERSION = 1

account
Kontoverbindung international

Type *fints.formals.KTI1*

all_accounts
Alle Konten

Type bool

date_end
Bis Datum

Type datetime.date

date_start
Von Datum

Type datetime.date

max_number_responses
Maximale Anzahl Einträge

Type int

supported_camt_messages
Kontoverbindung international

Type *fints.formals.SupportedMessageTypes*

touchdown_point
Aufsetzpunkt

Type str

class fints.segments.statement.**HKKAZ5**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Kontoumsätze anfordern/Zeitraum, version 5

Source: HBCI Homebanking-Computer-Interface, Schnittstellenspezifikation

TYPE = 'HKKAZ'

VERSION = 5

account
Kontoverbindung Auftraggeber

Type *fints.formals.Account2*

all_accounts
Alle Konten

Type bool

date_end
Bis Datum

Type datetime.date

date_start
Von Datum

Type datetime.date

max_number_responses
Maximale Anzahl Einträge

Type int

touchdown_point
Aufsetzpunkt

Type str

class fints.segments.statement.**HKKAZ6**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Kontoumsätze anfordern/Zeitraum, version 6

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HKKAZ'

VERSION = 6

account
Kontoverbindung Auftraggeber

Type *fints.formals.Account3*

all_accounts
Alle Konten

Type bool

date_end
Bis Datum

Type datetime.date

date_start
Von Datum

Type datetime.date

max_number_responses
Maximale Anzahl Einträge

Type int

touchdown_point
Aufsetzpunkt

Type str

class fints.segments.statement.**HKKAZ7**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

Kontoumsätze anfordern/Zeitraum, version 7

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HKKAZ'

VERSION = 7

account
Kontoverbindung international

Type *fints.formals.KTI1*

all_accounts
Alle Konten

Type bool

date_end
Bis Datum

Type datetime.date

date_start
Von Datum

Type datetime.date

max_number_responses
Maximale Anzahl Einträge

Type int

touchdown_point
Aufsetzpunkt

Type str

fints.segments.transfer module

class fints.segments.transfer.**HICCMS1**(**args*, ***kwargs*)
Bases: *fints.segments.base.ParameterSegment*

SEPA-Sammelüberweisung Parameter, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HICCMS'

VERSION = 1

parameter
Parameter SEPA-Sammelüberweisung

Type *fints.formals.BatchTransferParameter1*

class fints.segments.transfer.**HKCCM1**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

SEPA-Sammelüberweisung, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HKCCM'

VERSION = 1

account
Kontoverbindung international

Type *fints.formals.KTI1*

request_single_booking
Einzelbuchung gewünscht

Type bool

sepa_descriptor
SEPA Descriptor

Type str

sepa_pain_message
SEPA pain message

Type bytes

sum_amount
Summenfeld

Type *fints.formals.Amount1*

class fints.segments.transfer.**HKCCS1**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

SEPA Einzelüberweisung, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HKCCS'

VERSION = 1

account
Kontoverbindung international

Type *fints.formals.KTI1*

sepa_descriptor
SEPA Descriptor

Type str

sepa_pain_message
SEPA pain message

Type bytes

class fints.segments.transfer.**HKIPM1**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

SEPA-instant Sammelüberweisung, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HKIPM'

VERSION = 1

account
Kontoverbindung international

Type *fints.formals.KTI1*

request_single_booking
Einzelbuchung gewünscht

Type bool

sepa_descriptor
SEPA Descriptor

Type str

sepa_pain_message
SEPA pain message

Type bytes

sum_amount
Summenfeld

Type *fints.formals.Amount1*

class fints.segments.transfer.**HKIPZ1**(**args*, ***kwargs*)
Bases: *fints.segments.base.FinTS3Segment*

SEPA-instant Einzelüberweisung, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

TYPE = 'HKIPZ'

VERSION = 1

account
Kontoverbindung international

Type *fints.formals.KTI1*

sepa_descriptor
SEPA Descriptor

Type str

sepa_pain_message
SEPA pain message

Type bytes

2.1.3 FinTS Segment Sequence

A message is a sequence of segments. The SegmentSequence object allows searching for segments by type and version, by default recursing into nested sequences.

class fints.types.**SegmentSequence**(*segments=None*)
A sequence of FinTS3Segment objects

find_segment_first(**args*, ***kwargs*)
Finds the first matching segment.

Same parameters as find_segments(), but only returns the first match, or None if no match is found.

find_segment_highest_version(*query=None*, *version=None*, *callback=None*, *recurse=True*, *default=None*)

Finds the highest matching segment.

Same parameters as find_segments(), but returns the match with the highest version, or default if no match is found.

find_segments(*query=None*, *version=None*, *callback=None*, *recurse=True*, *throw=False*)

Yields an iterable of all matching segments.

Parameters

- **query** – Either a str or class specifying a segment type (such as 'HNHBK', or *HNHBK3*), or a list or tuple of strings or classes. If a list/tuple is specified, segments returning any matching type will be returned.
- **version** – Either an int specifying a segment version, or a list or tuple of ints. If a list/tuple is specified, segments returning any matching version will be returned.
- **callback** – A callable that will be given the segment as its sole argument and must return a boolean indicating whether to return this segment.
- **recurse** – If True (the default), recurse into SegmentSequenceField values, otherwise only look at segments in this SegmentSequence.
- **throw** – If True, a FinTSNoResponseError is thrown if no result is found. Defaults to False.

The match results of all given parameters will be AND-combined.

print_nested(*stream=None*, *level=0*, *indent=' '*, *prefix=''*, *first_level_indent=True*, *trailer=''*, *print_doc=True*, *first_line_suffix=''*)

render_bytes() → bytes

2.1.4 Working with Segments

Objects of *FinTS3Segment* or a subclass can be created by calling their constructor. The constructor takes optional arguments for all fields of the class. Setting and getting fields and subfields works, and consumes and returns Python objects as appropriate:

```
>>> from fints.segments import HNHBS1
>>> s = HNHBS1()
>>> s
fints.segments.HNHBS1(header=fints.formals.SegmentHeader('HNHBS', None, 1), message_
↪number=None)
>>> s.header.number = 3
>>> s.header
fints.formals.SegmentHeader('HNHBS', 3, 1)
```

When setting a value, format and length restrictions will be checked, if possible:

```
>>> s.message_number = 'abc'
ValueError: invalid literal for int() with base 10: 'abc'
>>> s.message_number = 12345
ValueError: Value '12345' cannot be rendered: max_length=4 exceeded
```

The only exception is: Every field can be set to `None` in order to clear the field and make it unset, recursively. No checking is performed whether all fields that are required (or conditionally required) by the specification are set. For

convenience, an unset constructed field will still be filled with an instance of the field's value type, so that subfield accessing will always work, without encountering `None` values on the way.

```
>>> s.header = None
>>> s
fints.segments.HNHBS1(header=fints.formals.SegmentHeader(None, None, None), message_
↪number=None)
```

When calling the constructor with non-keyword arguments, fields are assigned in order, with the exception of `header` in *FinTS3Segment* subclasses, which can only be given as a keyword argument. When no `header` argument is present, a *SegmentHeader* is automatically constructed with default values (and no `number`). It's generally not required to construct the `header` parameter manually.

```
>>> HNHBS1(42)
fints.segments.HNHBS1(header=fints.formals.SegmentHeader('HNHBS', None, 1), message_
↪number=42)
>>> HNHBS1(42, header=SegmentHeader('FOO'))
fints.segments.HNHBS1(header=fints.formals.SegmentHeader('FOO', None, None), message_
↪number=42)
```

Some segment fields have a variable number of values. These are always treated as a list, and minimum/maximum list length is obeyed. Setting a value beyond the end of the list results in an exception. Empty values are added to maintain the correct minimum number of values.

```
>>> from fints.segments import HIRMG2
>>> s = HIRMG2()
>>> s
fints.segments.HIRMG2(header=fints.formals.SegmentHeader('HIRMG', None, 2),␣
↪responses=[fints.formals.Response(code=None, reference_element=None, text=None)])
>>> s.responses[0].code = '0010'
>>> s.responses[1].code = '0100'
>>> s.print_nested()
fints.segments.HIRMG2(
    header = fints.formals.SegmentHeader('HIRMG', None, 2),
    responses = [
                fints.formals.Response(
                    code = '0010',
                    reference_element = None,
                    text = None,
                ),
                fints.formals.Response(
                    code = '0100',
                    reference_element = None,
                    text = None,
                ),
        ],
)
>>> HIRMG2(responses=[fints.formals.Response('2342')]).print_nested()
fints.segments.HIRMG2(
    header = fints.formals.SegmentHeader('HIRMG', None, 2),
    responses = [
                fints.formals.Response(
                    code = '2342',
                    reference_element = None,
                    text = None,
                ),
        ],
)
```

2.1.5 Defining new Segment classes

Base types

class fints.types.**Field**(*length=None*, *min_length=None*, *max_length=None*, *count=None*, *min_count=None*, *max_count=None*, *required=True*, *_d=None*)

render(*value*)

class fints.types.**TypedField**(*type=None*, **args*, ***kwargs*)

class fints.types.**ValueList**(*parent*)

class fints.types.**ContainerMeta**

class fints.types.**Container**(**args*, ***kwargs*)

classmethod naive_parse(*data*)

is_unset()

Field types

class fints.fields.**DataElementField**(**args*, ***kwargs*)

class fints.fields.**ContainerField**(*type=None*, **args*, ***kwargs*)

class fints.fields.**DataElementGroupField**(**args*, ***kwargs*)

class fints.fields.**GenericField**(**args*, ***kwargs*)

type = None

class fints.fields.**GenericGroupField**(**args*, ***kwargs*)

type = None

class fints.fields.**TextField**(**args*, ***kwargs*)

type = 'txt'

class fints.fields.**AlphanumericField**(**args*, ***kwargs*)

type = 'an'

class fints.fields.**DTAUSField**(**args*, ***kwargs*)

type = 'dta'

class fints.fields.**NumericField**(**args*, ***kwargs*)

type = 'num'

class fints.fields.**ZeroPaddedNumericField**(**args*, ***kwargs*)

type = ''

class fints.fields.**DigitsField**(**args*, ***kwargs*)

    type = 'dig'

class fints.fields.**FloatField**(**args*, ***kwargs*)

    type = 'float'

class fints.fields.**AmountField**(**args*, ***kwargs*)

    type = 'wrt'

class fints.fields.**BinaryField**(**args*, ***kwargs*)

    type = 'bin'

class fints.fields.**IDField**(**args*, ***kwargs*)

    type = 'id'

class fints.fields.**BooleanField**(**args*, ***kwargs*)

    type = 'jn'

class fints.fields.**CodeFieldMixin**(*enum=None*, **args*, ***kwargs*)

class fints.fields.**CodeField**(*enum=None*, **args*, ***kwargs*)

    type = 'code'

class fints.fields.**IntCodeField**(*enum=None*, **args*, ***kwargs*)

    type = ''

class fints.fields.**CountryField**(**args*, ***kwargs*)

    type = 'ctr'

class fints.fields.**CurrencyField**(**args*, ***kwargs*)

    type = 'cur'

class fints.fields.**DateField**(**args*, ***kwargs*)

    type = 'dat'

class fints.fields.**TimeField**(**args*, ***kwargs*)

    type = 'tim'

class fints.fields.**PasswordField**(**args*, ***kwargs*)

    type = ''

class fints.fields.**SegmentSequenceField**(**args*, ***kwargs*)

type = 'sf'

Constructed and helper types

class fints.formals.**DataElementGroup**(**args*, ***kwargs*)

class fints.formals.**SegmentHeader**(**args*, ***kwargs*)
Segmentkopf

type
Segmentkennung

Type str

number
Segmentnummer

Type int

version
Segmentversion

Type int

reference
Bezugssegment

Type int

class fints.formals.**ReferenceMessage**(**args*, ***kwargs*)

dialog_id

Type str

message_number

Type int

class fints.formals.**SecurityMethod**(**args*, ***kwargs*)
An enumeration.

DDV = 'DDV'

RAH = 'RAH'

RDH = 'RDH'

PIN = 'PIN'

class fints.formals.**SecurityProfile**(**args*, ***kwargs*)
Sicherheitsprofil

security_method
Sicherheitsverfahren

Type *fints.formals.SecurityMethod*

security_method_version
Version des Sicherheitsverfahrens

Type int

class fints.formals.**IdentifiedRole**(**args*, ***kwargs*)
An enumeration.

MS = '1'
Message Sender

MR = '2'
Message Receiver

class fints.formals.**SecurityIdentificationDetails**(**args*, ***kwargs*)

identified_role

Type *fints.formals.IdentifiedRole*

cid

Type bytes

identifier

Type str

class fints.formals.**DateTimeType**(**args*, ***kwargs*)
An enumeration.

STS = '1'
Sicherheitszeitstempel

CRT = '6'
Certificate Revocation Time

class fints.formals.**SecurityDateTime**(**args*, ***kwargs*)

date_time_type

Type *fints.formals.DateTimeType*

date

Type datetime.date

time

Type datetime.time

class fints.formals.**UsageEncryption**(**args*, ***kwargs*)
An enumeration.

OSY = '2'
Owner Symmetric

class fints.formals.**OperationMode**(**args*, ***kwargs*)
An enumeration.

CBC = '2'
Cipher Block Chaining

ISO_9796_1 = '16'
ISO 9796-1 (bei RDH)

ISO_9796_2_RANDOM = '17'
ISO 9796-2 mit Zufallszahl (bei RDH)

PKCS1V15 = '18'
RSASSA-PKCS#1 V1.5 (bei RDH); RSAES-PKCS#1 V1.5 (bei RAH, RDH)

PSS = '19'
RSASSA-PSS (bei RAH, RDH)

ZZZ = '999'
Gegenseitig vereinbart (DDV: Retail-MAC)

class fints.formals.**EncryptionAlgorithmCoded**(**args*, ***kwargs*)
An enumeration.

TWOKEY3DES = '13'
2-Key-Triple-DES

AES256 = '14'
AES-256

class fints.formals.**AlgorithmParameterName**(**args*, ***kwargs*)
An enumeration.

KYE = '5'
Symmetrischer Schlüssel, verschlüsselt mit symmetrischem Schlüssel

KYP = '6'
Symmetrischer Schlüssel, verschlüsselt mit öffentlichem Schlüssel

class fints.formals.**AlgorithmParameterIVName**(**args*, ***kwargs*)
An enumeration.

IVC = '1'
Initialization value, clear text

class fints.formals.**EncryptionAlgorithm**(**args*, ***kwargs*)

usage_encryption

Type *fints.formals.UsageEncryption*

operation_mode

Type *fints.formals.OperationMode*

encryption_algorithm

Type *fints.formals.EncryptionAlgorithmCoded*

algorithm_parameter_value

Type bytes

algorithm_parameter_name

Type *fints.formals.AlgorithmParameterName*

algorithm_parameter_iv_name

Type *fints.formals.AlgorithmParameterIVName*

algorithm_parameter_iv_value

Type bytes

class fints.formals.**HashAlgorithm**(**args*, ***kwargs*)

usage_hash

Type str

hash_algorithm

Type str

algorithm_parameter_name

Type str

algorithm_parameter_value

Type bytes

class fints.formals.**SignatureAlgorithm**(**args*, ***kwargs*)

usage_signature

Type str

signature_algorithm

Type str

operation_mode

Type str

class fints.formals.**BankIdentifier**(**args*, ***kwargs*)

COUNTRY_ALPHA_TO_NUMERIC = {'AT': '040', 'BE': '056', 'BG': '100', 'CA': '124', 'CH':

COUNTRY_NUMERIC_TO_ALPHA = {'040': 'AT', '056': 'BE', '100': 'BG', '124': 'CA', '1

country_identifier

Type str

bank_code

Type str

class fints.formals.**KeyType**(**args*, ***kwargs*)

Schlüsselart

D = 'D'

Schlüssel zur Erzeugung digitaler Signaturen

S = 'S'

Signierschlüssel

V = 'V'

Chiffrierschlüssel

class fints.formals.**KeyName**(**args*, ***kwargs*)

bank_identifier

Type *fints.formals.BankIdentifier*

user_id

Type str

key_type
Schlüsselart

Type *fints.formals.KeyType*

key_number

Type int

key_version

Type int

class fints.formals.**Certificate**(**args*, ***kwargs*)

certificate_type

Type str

certificate_content

Type bytes

class fints.formals.**UserDefinedSignature**(**args*, ***kwargs*)

pin

Type fints.utils.Password

tan

Type str

class fints.formals.**Response**(**args*, ***kwargs*)

code

Type str

reference_element

Type str

text

Type str

parameters

Type str

class fints.formals.**Amount1**(**args*, ***kwargs*)
Betrag

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

amount
Wert

Type decimal.Decimal

currency
Währung

Type str

class fints.formals.**AccountInformation**(**args*, ***kwargs*)

account_number

Type str

subaccount_number

Type str

bank_identifier

Type *fints.formals.BankIdentifier*

class fints.formals.**AccountLimit**(**args*, ***kwargs*)

limit_type

Type str

limit_amount

Type *fints.formals.Amount1*

limit_days

Type int

class fints.formals.**AllowedTransaction**(**args*, ***kwargs*)

transaction

Type str

required_signatures

Type int

limit_type

Type str

limit_amount

Type *fints.formals.Amount1*

limit_days

Type int

class fints.formals.**TANTimeDialogAssociation**(**args*, ***kwargs*)

An enumeration.

NOT_ALLOWED = '1'

TAN nicht zeitversetzt / dialogübergreifend erlaubt

ALLOWED = '2'

TAN zeitversetzt / dialogübergreifend erlaubt

BOTH = '3'

beide Verfahren unterstützt

NOT_APPLICABLE = '4'

nicht zutreffend

class fints.formals.**AllowedFormat**(**args*, ***kwargs*)
An enumeration.

NUMERIC = '1'
numerisch

ALPHANUMERIC = '2'
alfanumerisch

class fints.formals.**TANListNumberRequired**(**args*, ***kwargs*)
An enumeration.

NO = '0'
Nein

YES = '2'
Ja

class fints.formals.**InitializationMode**(**args*, ***kwargs*)
An enumeration.

CLEARTEXT_PIN_NO_TAN = '00'
Initialisierungsverfahren mit Klartext-PIN und ohne TAN

ENCRYPTED_PIN_NO_TAN = '01'
Schablone 01: Verschlüsselte PIN und ohne TAN

MASK_02 = '02'
Schablone 02: Reserviert, bei FinTS zur Zeit nicht verwendet

class fints.formals.**DescriptionRequired**(**args*, ***kwargs*)
An enumeration.

MUST_NOT = '0'
Bezeichnung des TAN-Mediums darf nicht angegeben werden

MAY = '1'
Bezeichnung des TAN-Mediums kann angegeben werden

MUST = '2'
Bezeichnung des TAN-Mediums muss angegeben werden

class fints.formals.**SMSChargeAccountRequired**(**args*, ***kwargs*)
An enumeration.

MUST_NOT = '0'
SMS-Abbuchungskonto darf nicht angegeben werden

MAY = '1'
SMS-Abbuchungskonto kann angegeben werden

MUST = '2'
SMS-Abbuchungskonto muss angegeben werden

class fints.formals.**PrincipalAccountRequired**(**args*, ***kwargs*)
An enumeration.

MUST_NOT = '0'
Auftraggeberkonto darf nicht angegeben werden

MUST = '2'
Auftraggeberkonto muss angegeben werden, wenn im Geschäftsvorfall enthalten

class fints.formals.**TaskHashAlgorithm**(**args*, ***kwargs*)
An enumeration.

NONE = '0'
Auftrags-Hashwert nicht unterstützt

RIPEMD_160 = '1'
RIPEMD-160

SHA_1 = '2'
SHA-1

class fints.formals.**TwoStepParametersCommon**(**args*, ***kwargs*)

VERSION
TAN mechanism version

security_function
Sicherheitsfunktion kodiert

Type str

tan_process
TAN-Prozess

Type str

tech_id
Technische Identifikation TAN-Verfahren

Type str

class fints.formals.**TwoStepParameters1**(**args*, ***kwargs*)

name
Name des Zwei-Schritt-Verfahrens

Type str

max_length_input
Maximale Länge des Eingabewertes im Zwei-Schritt-Verfahren

Type int

allowed_format
Erlaubtes Format im Zwei-Schritt-Verfahren

Type *fints.formals.AllowedFormat*

text_return_value
Text zur Belegung des Rückgabewertes im Zwei-Schritt-Verfahren

Type str

max_length_return_value
Maximale Länge des Rückgabewertes im Zwei-Schritt-Verfahren

Type int

number_of_supported_lists
Anzahl unterstützter aktiver TAN-Listen

Type int

multiple_tans_allowed
Mehrfach-TAN erlaubt

Type bool

tan_time_delayed_allowed
TAN zeitversetzt/dialogübergreifend erlaubt

Type bool

class fints.formals.**TwoStepParameters2**(**args*, ***kwargs*)

name
Name des Zwei-Schritt-Verfahrens

Type str

max_length_input
Maximale Länge des Eingabewertes im Zwei-Schritt-Verfahren

Type int

allowed_format
Erlaubtes Format im Zwei-Schritt-Verfahren

Type *fints.formals.AllowedFormat*

text_return_value
Text zur Belegung des Rückgabewertes im Zwei-Schritt-Verfahren

Type str

max_length_return_value
Maximale Länge des Rückgabewertes im Zwei-Schritt-Verfahren

Type int

number_of_supported_lists
Anzahl unterstützter aktiver TAN-Listen

Type int

multiple_tans_allowed
Mehrfach-TAN erlaubt

Type bool

tan_time_dialog_association
TAN Zeit- und Dialogbezug

Type *fints.formals.TANTimeDialogAssociation*

tan_list_number_required
TAN-Listennummer erforderlich

Type *fints.formals.TANListNumberRequired*

cancel_allowed
Auftragsstorno erlaubt

Type bool

challenge_class_required
Challenge-Klasse erforderlich

Type bool

challenge_value_required
Challenge-Betrag erforderlich

Type bool

class fints.formals.**TwoStepParameters3**(**args*, ***kwargs*)

name
Name des Zwei-Schritt-Verfahrens

Type str

max_length_input
Maximale Länge des Eingabewertes im Zwei-Schritt-Verfahren

Type int

allowed_format
Erlaubtes Format im Zwei-Schritt-Verfahren

Type *fints.formals.AllowedFormat*

text_return_value
Text zur Belegung des Rückgabewertes im Zwei-Schritt-Verfahren

Type str

max_length_return_value
Maximale Länge des Rückgabewertes im Zwei-Schritt-Verfahren

Type int

number_of_supported_lists
Anzahl unterstützter aktiver TAN-Listen

Type int

multiple_tans_allowed
Mehrfach-TAN erlaubt

Type bool

tan_time_dialog_association
TAN Zeit- und Dialogbezug

Type *fints.formals.TANTimeDialogAssociation*

tan_list_number_required
TAN-Listennummer erforderlich

Type *fints.formals.TANListNumberRequired*

cancel_allowed
Auftragsstorno erlaubt

Type bool

challenge_class_required
Challenge-Klasse erforderlich

Type bool

challenge_value_required
Challenge-Betrag erforderlich

Type bool

initialization_mode
Initialisierungsmodus

Type *fints.formals.InitializationMode*

description_required
Bezeichnung des TAN-Medium erforderlich

Type *fints.formals.DescriptionRequired*

supported_media_number
Anzahl unterstützter aktiver TAN-Medien

Type int

class fints.formals.**TwoStepParameters4**(**args*, ***kwargs*)

zka_id
ZKA TAN-Verfahren

Type str

zka_version
Version ZKA TAN-Verfahren

Type str

name
Name des Zwei-Schritt-Verfahrens

Type str

max_length_input
Maximale Länge des Eingabewertes im Zwei-Schritt-Verfahren

Type int

allowed_format
Erlaubtes Format im Zwei-Schritt-Verfahren

Type *fints.formals.AllowedFormat*

text_return_value
Text zur Belegung des Rückgabewertes im Zwei-Schritt-Verfahren

Type str

max_length_return_value
Maximale Länge des Rückgabewertes im Zwei-Schritt-Verfahren

Type int

number_of_supported_lists
Anzahl unterstützter aktiver TAN-Listen

Type int

multiple_tans_allowed
Mehrfach-TAN erlaubt

Type bool

tan_time_dialog_association
TAN Zeit- und Dialogbezug

Type *fints.formals.TANTimeDialogAssociation*

tan_list_number_required
TAN-Listennummer erforderlich

Type *fints.formals.TANListNumberRequired*

cancel_allowed
Auftragsstorno erlaubt

Type bool

sms_charge_account_required
SMS-Abbuchungskonto erforderlich

Type bool

challenge_class_required
Challenge-Klasse erforderlich

Type bool

challenge_value_required
Challenge-Betrag erforderlich

Type bool

challenge_structured
Challenge strukturiert

Type bool

initialization_mode
Initialisierungsmodus

Type *fints.formals.InitializationMode*

description_required
Bezeichnung des TAN-Medium erforderlich

Type *fints.formals.DescriptionRequired*

supported_media_number
Anzahl unterstützter aktiver TAN-Medien

Type int

class fints.formals.**TwoStepParameters5**(**args*, ***kwargs*)

zka_id
ZKA TAN-Verfahren

Type str

zka_version
Version ZKA TAN-Verfahren

Type str

name
Name des Zwei-Schritt-Verfahrens

Type str

max_length_input
Maximale Länge des Eingabewertes im Zwei-Schritt-Verfahren

Type int

allowed_format
Erlaubtes Format im Zwei-Schritt-Verfahren

Type *fints.formals.AllowedFormat*

text_return_value
Text zur Belegung des Rückgabewertes im Zwei-Schritt-Verfahren

Type str

max_length_return_value
Maximale Länge des Rückgabewertes im Zwei-Schritt-Verfahren

Type int

number_of_supported_lists
Anzahl unterstützter aktiver TAN-Listen

Type int

multiple_tans_allowed
Mehrfach-TAN erlaubt

Type bool

tan_time_dialog_association
TAN Zeit- und Dialogbezug

Type *fints.formals.TANTimeDialogAssociation*

tan_list_number_required
TAN-Listennummer erforderlich

Type *fints.formals.TANListNumberRequired*

cancel_allowed
Auftragsstorno erlaubt

Type bool

sms_charge_account_required
SMS-Abbuchungskonto erforderlich

Type *fints.formals.SMSChargeAccountRequired*

principal_account_required
Auftraggeberkonto erforderlich

Type *fints.formals.PrincipalAccountRequired*

challenge_class_required
Challenge-Klasse erforderlich

Type bool

challenge_structured
Challenge strukturiert

Type bool

initialization_mode
Initialisierungsmodus

Type *fints.formals.InitializationMode*

description_required
Bezeichnung des TAN-Medium erforderlich

Type *fints.formals.DescriptionRequired*

supported_media_number
Anzahl unterstützter aktiver TAN-Medien

Type int

class fints.formals.**TwoStepParameters6**(**args*, ***kwargs*)

zka_id
ZKA TAN-Verfahren

Type str

zka_version
Version ZKA TAN-Verfahren

Type str

name
Name des Zwei-Schritt-Verfahrens

Type str

max_length_input
Maximale Länge des Eingabewertes im Zwei-Schritt-Verfahren

Type int

allowed_format
Erlaubtes Format im Zwei-Schritt-Verfahren

Type *fints.formals.AllowedFormat*

text_return_value
Text zur Belegung des Rückgabewertes im Zwei-Schritt-Verfahren

Type str

max_length_return_value
Maximale Länge des Rückgabewertes im Zwei-Schritt-Verfahren

Type int

multiple_tans_allowed
Mehrfach-TAN erlaubt

Type bool

tan_time_dialog_association
TAN Zeit- und Dialogbezug

Type *fints.formals.TANTimeDialogAssociation*

cancel_allowed
Auftragsstorno erlaubt

Type bool

sms_charge_account_required
SMS-Abbuchungskonto erforderlich

Type *fints.formals.SMSChargeAccountRequired*

principal_account_required
Auftraggeberkonto erforderlich

Type *fints.formals.PrincipalAccountRequired*

challenge_class_required
Challenge-Klasse erforderlich

Type bool

challenge_structured
Challenge strukturiert

Type bool

initialization_mode
Initialisierungsmodus

Type *fints.formals.InitializationMode*

description_required
Bezeichnung des TAN-Medium erforderlich

Type *fints.formals.DescriptionRequired*

response_hhd_uc_required
Antwort HHD_UC erforderlich

Type bool

supported_media_number
Anzahl unterstützter aktiver TAN-Medien

Type int

class fints.formals.**ParameterTwostepCommon**(**args*, ***kwargs*)

onestep_method_allowed

Type bool

multiple_tasks_allowed

Type bool

task_hash_algorithm
Auftrags-Hashwertverfahren

Type *fints.formals.TaskHashAlgorithm*

class fints.formals.**ParameterTwostepTAN1**(**args*, ***kwargs*)

security_profile_bank_signature

Type str

twostep_parameters

Type *fints.formals.TwoStepParameters1*

class fints.formals.**ParameterTwostepTAN2**(**args*, ***kwargs*)

twostep_parameters

Type *fints.formals.TwoStepParameters2*

class fints.formals.**ParameterTwostepTAN3**(**args*, ***kwargs*)

twostep_parameters

Type *fints.formals.TwoStepParameters3*

class fints.formals.**ParameterTwostepTAN4**(**args*, ***kwargs*)

twostep_parameters

Type *fints.formals.TwoStepParameters4*

class fints.formals.**ParameterTwostepTAN5**(**args*, ***kwargs*)

twostep_parameters

Type *fints.formals.TwoStepParameters5*

class fints.formals.**ParameterTwostepTAN6**(**args*, ***kwargs*)

twostep_parameters

Type *fints.formals.TwoStepParameters6*

class fints.formals.**TransactionTanRequired**(**args*, ***kwargs*)

transaction

Type str

tan_required

Type bool

class fints.formals.**ParameterPinTan**(**args*, ***kwargs*)

min_pin_length

Type int

max_pin_length

Type int

max_tan_length

Type int

user_id_field_text

Type str

customer_id_field_text

Type str

transaction_tans_required

Type *fints.formals.TransactionTanRequired*

class fints.formals.**Language2**(**args*, ***kwargs*)

Dialogsprache

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Formals

DEFAULT = '0'
Standard

DE = '1'
Deutsch, 'de', Subset Deutsch, Codeset 1 (Latin 1)

EN = '2'
Englisch, 'en', Subset Englisch, Codeset 1 (Latin 1)

FR = '3'
Französisch, 'fr', Subset Französisch, Codeset 1 (Latin 1)

class fints.formals.**SupportedLanguages2**(**args*, ***kwargs*)

languages

Type *fints.formals.Language2*

class fints.formals.**SupportedHBCIVersions2**(**args*, ***kwargs*)

versions

Type str

class fints.formals.**KTZ1**(**args*, ***kwargs*)
Kontoverbindung ZV international, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

is_sepa
Kontoverwendung SEPA

Type bool

iban
IBAN

Type str

bic
BIC

Type str

account_number
Konto-/Depotnummer

Type str

subaccount_number
Unterkontomerkmal

Type str

bank_identifier
Kreditinstitutskennung

Type *fints.formals.BankIdentifier*

as_sepa_account()

classmethod from_sepa_account(*acc*)

class fints.formals.**KTI1**(**args*, ***kwargs*)
Kontoverbindung international, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

iban
IBAN

Type str

bic
BIC

Type str

account_number
Konto-/Depotnummer

Type str

subaccount_number
Unterkontomerkmal

Type str

bank_identifier
Kreditinstitutskennung

Type *fints.formals.BankIdentifier*

classmethod from_sepa_account(*acc*)

class fints.formals.**Account2**(**args*, ***kwargs*)
Kontoverbindung, version 2

Source: HBCI Homebanking-Computer-Interface, Schnittstellenspezifikation

account_number
Konto-/Depotnummer

Type str

subaccount_number
Unterkontomerkmal

Type str

country_identifier
Länderkennzeichen

Type str

bank_code
Kreditinstitutscode

Type str

classmethod from_sepa_account(*acc*)

class fints.formals.**Account3**(**args*, ***kwargs*)
Kontoverbindung, version 3

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

account_number
Konto-/Depotnummer

Type str

subaccount_number
Unterkontomerkmal

Type str

bank_identifier
Kreditinstitutskennung

Type *fints.formals.BankIdentifier*

classmethod from_sepa_account(*acc*)

class fints.formals.**SecurityRole**(**args*, ***kwargs*)
Rolle des Sicherheitslieferanten, kodiert, version 2

Kodierte Information über das Verhältnis desjenigen, der bezüglich der zu si-chernden Nachricht die Sicherheit gewährleistet. Die Wahl ist von der bankfachlichen Auslegung der Signatur, respektive vom vertraglichen Zustand zwischen Kunde und Kreditinstitut abhängig.

Source: FinTS Financial Transaction Services, Sicherheitsverfahren HBCI

ISS = '1'
Erfasser, Erstsignatur

CON = '3'
Unterstützer, Zweitsignatur

WIT = '4'
Zeuge/Übermittler, nicht Erfasser

class fints.formals.**CompressionFunction**(**args*, ***kwargs*)
Komprimierungsfunktion, version 2

Source: FinTS Financial Transaction Services, Sicherheitsverfahren HBCI

NULL = '0'
Keine Kompression

LZW = '1'
Lempel, Ziv, Welch

COM = '2'
Optimized LZW

LZSS = '3'
Lempel, Ziv

LZHuf = '4'
LZ + Huffman Coding

ZIP = '5'
PKZIP

GZIP = '6'
deflate (http://www.gzip.org/zlib)

BZIP2 = '7'
bzip2 (http://sourceware.cygnus.com/bzip2/)

ZZZ = '999'
Gegenseitig vereinbart

class fints.formals.**SecurityApplicationArea**(**args*, ***kwargs*)
Bereich der Sicherheitsapplikation, kodiert, version 2

Informationen darüber, welche Daten vom kryptographischen Prozess verarbeitet werden.

Source: FinTS Financial Transaction Services, Sicherheitsverfahren HBCI

SHM = '1'
Signaturkopf und HBCI-Nutzdaten

SHT = '2'
Von Signaturkopf bis Signaturabschluss

class fints.formals.**SecurityClass**(**args*, ***kwargs*)
Sicherheitsklasse, version 1

Die Sicherheitsklasse gibt für jede Signatur den erforderlichen Sicherheitsdienst an.

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Formals

NONE = 0
Kein Sicherheitsdienst erforderlich

AUTH = 1
Sicherheitsdienst 'Authentikation'

AUTH_ADV = 2
Sicherheitsdienst 'Authentikation' mit fortgeschrittener elektronischer Signatur, optionaler Zertifikatsprüfung

NON_REPUD = 3
Sicherheitsdienst 'Non-Repudiation' mit fortgeschrittener elektronischer Signatur, optionaler Zertifikatsprüfung

NON_REPUD_QUAL = 4
Sicherheitsdienst 'Non-Repudiation' mit fortgeschrittener bzw. qualifizierter elektronischer Signatur, zwingende Zertifikatsprüfung

class fints.formals.**UPDUsage**(**args*, ***kwargs*)
UPD-Verwendung, version 2

Kennzeichen dafür, wie diejenigen Geschäftsvorfälle zu interpretieren sind, die bei der Beschreibung der Kontoinformationen nicht unter den erlaubten Geschäftsvorfällen aufgeführt sind.

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Formals

UPD_CONCLUSIVE = '0'
Die nicht aufgeführten Geschäftsvorfälle sind gesperrt

UPD_INCONCLUSIVE = '1'
Bei nicht aufgeführten Geschäftsvorfällen ist keine Aussage möglich, ob diese erlaubt oder gesperrt sind

class fints.formals.**SystemIDStatus**(**args*, ***kwargs*)
Kundensystem-Status, version 2

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Formals

ID_UNNECESSARY = '0'
Kundensystem-ID wird nicht benötigt

ID_NECESSARY = '1'
Kundensystem-ID wird benötigt

class fints.formals.**SynchronizationMode**(**args*, ***kwargs*)
Synchronisierungsmodus, version 2

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Formals

NEW_SYSTEM_ID = '0'
Neue Kundensystem-ID zurückmelden

LAST_MESSAGE = '1'
Letzte verarbeitete Nachrichtennummer zurückmelden

SIGNATURE_ID = '2'
Signatur-ID zurückmelden

class fints.formals.**CreditDebit2**(**args*, ***kwargs*)
Soll-Haben-Kennzeichen, version 2

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

CREDIT = 'C'
Haben

DEBIT = 'D'
Soll

class fints.formals.**Balance1**(**args*, ***kwargs*)
Saldo, version 1

Source: HBCI Homebanking-Computer-Interface, Schnittstellenspezifikation

credit_debit
Soll-Haben-Kennzeichen

Type *fints.formals.CreditDebit2*

amount
Wert

Type decimal.Decimal

currency
Währung

Type str

date
Datum

Type datetime.date

time
Uhrzeit

Type datetime.time

as_mt940_Balance()

class fints.formals.**Balance2**(**args*, ***kwargs*)
Saldo, version 2

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

credit_debit
Soll-Haben-Kennzeichen

Type *fints.formals.CreditDebit2*

amount
Betrag

Type *fints.formals.Amount1*

date
Datum

Type datetime.date

time
Uhrzeit

Type datetime.time

as_mt940_Balance()

class fints.formals.**Timestamp1**(**args*, ***kwargs*)
Zeitstempel

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

date
Datum

Type datetime.date

time
Uhrzeit

Type datetime.time

class fints.formals.**TANMediaType2**(**args*, ***kwargs*)
TAN-Medium-Art

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

ALL = '0'
Alle

ACTIVE = '1'
Aktiv

AVAILABLE = '2'
Verfügbar

class fints.formals.**TANMediaClass3**(**args*, ***kwargs*)
TAN-Medium-Klasse, version 3

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

ALL = 'A'
Alle Medien

LIST = 'L'
Liste

GENERATOR = 'G'
TAN-Generator

MOBILE = 'M'
Mobiltelefon mit mobileTAN

SECODER = 'S'
Secoder

class fints.formals.**TANMediaClass4**(**args*, ***kwargs*)
TAN-Medium-Klasse, version 4

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

ALL = 'A'
Alle Medien

LIST = 'L'
Liste

GENERATOR = 'G'
TAN-Generator

MOBILE = 'M'
Mobiltelefon mit mobileTAN

SECODER = 'S'
Secoder

BILATERAL = 'B'
Bilateral vereinbart

class fints.formals.**TANMediumStatus**(**args*, ***kwargs*)
Status

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

ACTIVE = '1'
Aktiv

AVAILABLE = '2'
Verfügbar

ACTIVE_SUCCESSOR = '3'
Aktiv Folgekarte

AVAILABLE_SUCCESSOR = '4'
Verfügbar Folgekarte

class fints.formals.**TANMedia4**(**args*, ***kwargs*)
TAN-Medium-Liste, version 4

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

tan_medium_class
TAN-Medium-Klasse

Type *fints.formals.TANMediaClass3*

status
Status

Type *fints.formals.TANMediumStatus*

card_number
Kartennummer

Type str

card_sequence
Kartenfolgenummer

Type str

card_type
Kartenart

Type int

account
Kontonummer Auftraggeber

Type *fints.formals.Account3*

valid_from
Gültig ab

Type datetime.date

valid_until
Gültig bis

Type datetime.date

tan_list_number
TAN-Listennummer

Type str

tan_medium_name
Bezeichnung des TAN-Mediums

Type str

mobile_number_masked
Mobiltelefonnummer, verschleiert

Type str

mobile_number
Mobiltelefonnummer

Type str

sms_charge_account
SMS-Abbuchungskonto

Type *fints.formals.KTI1*

number_free_tans
Anzahl freie TANs

Type int

last_use
Letzte Benutzung

Type datetime.date

active_since
Freigeschaltet am

Type datetime.date

class fints.formals.**TANMedia5**(**args*, ***kwargs*)
TAN-Medium-Liste, version 5

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

tan_medium_class
TAN-Medium-Klasse

Type *fints.formals.TANMediaClass4*

status
Status

Type *fints.formals.TANMediumStatus*

security_function
Sicherheitsfunktion, kodiert

Type int

card_number
Kartennummer

Type str

card_sequence
Kartenfolgenummer

Type str

card_type
Kartenart

Type int

account
Kontonummer Auftraggeber

Type *fints.formals.Account3*

valid_from
Gültig ab

Type datetime.date

valid_until
Gültig bis

Type datetime.date

tan_list_number
TAN-Listennummer

Type str

tan_medium_name
Bezeichnung des TAN-Mediums

Type str

mobile_number_masked
Mobiltelefonnummer, verschleiert

Type str

mobile_number
Mobiltelefonnummer

Type str

sms_charge_account
SMS-Abbuchungskonto

Type *fints.formals.KTI1*

number_free_tans
Anzahl freie TANs

Type int

last_use
Letzte Benutzung

Type datetime.date

active_since
Freigeschaltet am

Type datetime.date

class fints.formals.**TANUsageOption**(**args*, ***kwargs*)
TAN-Einsatzoption

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

ALL_ACTIVE = '0'
Kunde kann alle "aktiven" Medien parallel nutzen

EXACTLY_ONE = '1'
Kunde kann genau ein Medium zu einer Zeit nutzen

MOBILE_AND_GENERATOR = '2'
Kunde kann ein Mobiltelefon und einen TAN-Generator parallel nutzen

class fints.formals.**ParameterChallengeClass**(**args*, ***kwargs*)
Parameter Challenge-Klasse

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

parameters

Type str

class fints.formals.**ResponseHHDUC**(**args*, ***kwargs*)
Antwort HHD_UC

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

atc
ATC

Type str

ac
Application Cryptogram AC

Type bytes

ef_id_data
EF_ID Data

Type bytes

cvr
CVR

Type bytes

version_info_chiptan
Versionsinfo der chipTAN-Applikation

Type bytes

class fints.formals.**ChallengeValidUntil**(**args*, ***kwargs*)
Gültigkeitsdatum und -uhrzeit für Challenge

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Sicherheitsverfahren PIN/TAN

date
Datum

Type datetime.date

time
Uhrzeit

Type datetime.time

class fints.formals.**BatchTransferParameter1**(**args*, ***kwargs*)
Parameter SEPA-Sammelüberweisung, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

max_transfer_count
Maximale Anzahl CreditTransferTransactionInformation

Type int

sum_amount_required
Summenfeld benötigt

Type bool

single_booking_allowed
Einzelbuchung erlaubt

Type bool

class fints.formals.**ServiceType2**(**args*, ***kwargs*)
An enumeration.

T_ONLINE = 1
T-Online

TCP_IP = 2
TCP/IP (Protokollstack SLIP/PPP)

HTTPS = 3
https

class fints.formals.**CommunicationParameter2**(**args*, ***kwargs*)
Kommunikationsparameter, version 2

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Formals

service_type
Kommunikationsdienst

Type *fints.formals.ServiceType2*

address
Kommunikationsadresse

Type str

address_adjunct
Kommunikationsadresszusatz

Type str

filter_function
Filterfunktion

Type str

filter_function_version
Version der Filterfunktion

Type int

class fints.formals.**ScheduledDebitParameter1**(**args*, ***kwargs*)
Parameter terminierte SEPA-Einzellastschrift einreichen, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

min_advance_notice_FNAL_RCUR
Minimale Vorlaufzeit FNAL/RCUR

Type int

max_advance_notice_FNAL_RCUR
Maximale Vorlaufzeit FNAL/RCUR

Type int

min_advance_notice_FRST_OOFF
Minimale Vorlaufzeit FRST/OOFF

Type int

max_advance_notice_FRST_OOFF
Maximale Vorlaufzeit FRST/OOFF

Type int

class fints.formals.**ScheduledDebitParameter2**(**args*, ***kwargs*)
Parameter terminierte SEPA-Einzellastschrift einreichen, version 2

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

min_advance_notice
Minimale Vorlaufzeit SEPA-Lastschrift

Type str

max_advance_notice
Maximale Vorlaufzeit SEPA-Lastschrift

Type str

allowed_purpose_codes
Zulässige purpose codes

Type str

supported_sepa_formats
Unterstützte SEPA-Datenformate

Type str

class fints.formals.**ScheduledBatchDebitParameter1**(**args*, ***kwargs*)
Parameter terminierte SEPA-Sammellastschrift einreichen, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

min_advance_notice_FNAL_RCUR
Minimale Vorlaufzeit FNAL/RCUR

Type int

max_advance_notice_FNAL_RCUR
Maximale Vorlaufzeit FNAL/RCUR

Type int

min_advance_notice_FRST_OOFF
Minimale Vorlaufzeit FRST/OOFF

Type int

max_advance_notice_FRST_OOFF
Maximale Vorlaufzeit FRST/OOFF

Type int

max_debit_count
Maximale Anzahl DirectDebitTransfer TransactionInformation

Type int

sum_amount_required
Summenfeld benötigt

Type bool

single_booking_allowed
Einzelbuchung erlaubt

Type bool

class fints.formals.**ScheduledBatchDebitParameter2**(**args*, ***kwargs*)
Parameter terminierte SEPA-Sammellastschrift einreichen, version 2

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

min_advance_notice
Minimale Vorlaufzeit SEPA-Lastschrift

Type str

max_advance_notice
Maximale Vorlaufzeit SEPA-Lastschrift

Type str

max_debit_count
Maximale Anzahl DirectDebitTransfer TransactionInformation

Type int

sum_amount_required
Summenfeld benötigt

Type bool

single_booking_allowed
Einzelbuchung erlaubt

Type bool

allowed_purpose_codes
Zulässige purpose codes

Type str

supported_sepa_formats
Unterstützte SEPA-Datenformate

Type str

class fints.formals.**ScheduledCOR1DebitParameter1**(**args*, ***kwargs*)
Parameter terminierte SEPA-COR1-Einzellastschrift, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

min_advance_notice_FNAL_RCUR
Minimale Vorlaufzeit FNAL/RCUR

Type int

max_advance_notice_FNAL_RCUR
Maximale Vorlaufzeit FNAL/RCUR

Type int

min_advance_notice_FRST_OOFF
Minimale Vorlaufzeit FRST/OOFF

Type int

max_advance_notice_FRST_OOFF
Maximale Vorlaufzeit FRST/OOFF

Type int

allowed_purpose_codes
Zulässige purpose codes

Type str

supported_sepa_formats
Unterstützte SEPA-Datenformate

Type str

class fints.formals.**ScheduledCOR1BatchDebitParameter1**(**args*, ***kwargs*)
Parameter terminierte SEPA-COR1-Sammellastschrift, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

max_debit_count
Maximale Anzahl DirectDebitTransfer TransactionInformation

Type int

sum_amount_required
Summenfeld benötigt

Type bool

single_booking_allowed
Einzelbuchung erlaubt

Type bool

min_advance_notice_FNAL_RCUR
Minimale Vorlaufzeit FNAL/RCUR

Type int

max_advance_notice_FNAL_RCUR
Maximale Vorlaufzeit FNAL/RCUR

Type int

min_advance_notice_FRST_OOFF
Minimale Vorlaufzeit FRST/OOFF

Type int

max_advance_notice_FRST_OOFF
Maximale Vorlaufzeit FRST/OOFF

Type int

allowed_purpose_codes
Zulässige purpose codes

Type str

supported_sepa_formats
Unterstützte SEPA-Datenformate

Type str

class fints.formals.**SupportedSEPAPainMessages1**(**args*, ***kwargs*)
Unterstützte SEPA pain messages, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

sepa_descriptors
SEPA Descriptor

Type str

class fints.formals.**QueryScheduledDebitParameter1**(**args*, ***kwargs*)
Parameter Bestand terminierter SEPA-Einzellastschriften, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

date_range_allowed
Zeitraum möglich

Type bool

max_number_responses_allowed
Eingabe Anzahl Einträge erlaubt

Type bool

class fints.formals.**QueryScheduledDebitParameter2**(**args*, ***kwargs*)
Parameter Bestand terminierter SEPA-Einzellastschriften, version 2

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

max_number_responses_allowed
Eingabe Anzahl Einträge erlaubt

Type bool

date_range_allowed
Zeitraum möglich

Type bool

supported_sepa_formats
Unterstützte SEPA-Datenformate

Type str

class fints.formals.**QueryScheduledBatchDebitParameter1**(**args*, ***kwargs*)
Parameter Bestand terminierter SEPA-Sammellastschriften, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

max_number_responses_allowed
Eingabe Anzahl Einträge erlaubt

Type bool

date_range_allowed
Zeitraum möglich

Type bool

class fints.formals.**QueryCreditCardStatements2**(**args*, ***kwargs*)
Parameter Kreditkartenumsätze anfordern, version 2

Source: reverse engineered

cutoff_days
Maximale Vorhaltezeit der Umsätze

Type int

max_number_responses_allowed
Eingabe Anzahl Einträge erlaubt

Type bool

date_range_allowed
Zeitraum möglich

Type bool

class fints.formals.**SEPACCode1**(**args*, ***kwargs*)
An enumeration.

REVERSAL = '1'
Reversal

REVOCATION = '2'
Revocation

DELETION = '3'
Delete

class fints.formals.**StatusSEPATask1**(**args*, ***kwargs*)
An enumeration.

PENDING = '1'
In Terminierung

DECLINED = '2'
Abgelehnt von erster Inkassostelle

IN_PROGRESS = '3'
in Bearbeitung

PROCESSED = '4'
Creditoren-seitig verarbeitet, Buchung veranlasst

REVOKED = '5'
R-Transaktion wurde veranlasst

class fints.formals.**GetSEPAAccountParameter1**(**args*, ***kwargs*)
Parameter SEPA-Kontoverbindung anfordern, version 1

Source: FinTS Financial Transaction Services, Schnittstellenspezifikation, Messages – Multibankfähige Geschäftsvorfälle

single_account_query_allowed
Einzelkontenabruf erlaubt

Type bool

national_account_allowed
Nationale Kontoverbindung erlaubt

Type bool

structured_purpose_allowed
Strukturierter Verwendungszweck erlaubt

Type bool

supported_sepa_formats
Unterstützte SEPA-Datenformate

Type str

class fints.formals.**SupportedMessageTypes**(**args*, ***kwargs*)
Unterstützte camt-Messages

Source: Messages - Multibankfähige Geschäftsvorfälle (SEPA) - C.2.3.1.1.1

expected_type
Unterstützte camt-messages

Type str

class fints.formals.**BookedCamtStatements1**(**args*, ***kwargs*)
Gebuchte camt-Umsätze

Source: Messages - Multibankfähige Geschäftsvorfälle (SEPA)

camt_statements
camt-Umsätze gebucht

Type bytes

Python Module Index

f

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Y

Z