SATComum Documentation

Release 2.2

Base4 Sistemas Ltda

nov. 21, 2021

Sumário

1 Projetos Relacionados **3**

2 Participe **5**

3 Documentação da API **7**

3.1 Módulo `satcomum.br` 7

3.2 Módulo `satcomum.constantes` 9

3.3 Módulo `satcomum.ersat` 9

3.4 Módulo `satcomum.util` 10

4 Tabelas e Índices **13**

Índice de Módulos Python **15**

Índice **17**

Este projeto mantém o código comum aos projetos relacionados ao SAT-CF-e, tais como validadores, formatadores, constantes e utilitários que são usados nos projetos relacionados.

Nesta documentação você encontrará basicamente a documentação da API para esse código comum. Para uma visão mais abrangente da tecnologia SAT-CF-e e de como estes projetos se relacionam conheça os projetos relacionados.

CAPÍTULO 1

Projetos Relacionados

Este projeto é apenas uma parte de um total de cinco projetos que compõem uma solução compreensível para a tecnologia SAT-CF-e em linguagem Python, disponíveis para integração nas aplicações de ponto-de-venda. São eles:

- **Projeto SATCFe** Abstração para acesso às funções da biblioteca SAT em Python. Visite a documentação desse projeto para uma visão mais abrangente da tecnologia SAT-CF-e e de como estes projetos se relacionam.
- **Projeto SATExtrato** Impressão dos extratos do CF-e-SAT. Este projeto é capaz de imprimir extratos de documentos de venda ou de cancelamento diretamente a partir dos documentos XML que os representam. A impressão tem um alto grau de compatibilidade com mini-impressoras (conhecidas como impressoras não-fiscais) já que é baseada na tecnologia Epson® ESC/POS™ através do projeto PyESCPOS.
- **Projeto SATHub** Torna possível o compartilhamento de equipamentos SAT com múltiplos pontos- de-venda, além de tornar possível que aplicações heterogêneas, escritas em outras linguagens de programação ou de outras plataformas, acessem o equipamento SAT.
- **Projeto PyESCPOS** Implementa o suporte à tecnologia Epson® ESC/POS™ compatível com a imensa maioria das mini-impressoras disponíveis no mercado.

CAPÍTULO 2

Participe

Participe deste projeto ou de qualquer um dos projetos relacionados. Se você puder contribuir com código, excelente! Faça um clone do repositório, modifique o que acha que deve e faça o *pull-request*. Teremos prazer em aceitar o seu código.

Se você não quer (ou não pode) programar, também pode contribuir com documentação. Ou ainda, se você vir algo errado ou achar que algo não está certo, conte pra gente criando um incidente na página do projeto.

Siga-nos no Github ou no Twitter.

CAPÍTULO 3

Documentação da API

3.1 Módulo satcomum.br

exception satcomum.br.**NumeroCEPError**

exception satcomum.br.**NumeroCNPJCPFError**

exception satcomum.br.**NumeroCNPJError**

exception satcomum.br.**NumeroCPFError**

exception satcomum.br.**UnidadeFederativaError**

satcomum.br.**as_cep**(*numero*)

Formata um número de CEP. Se o argumento não for um CEP válido apenas retorna o argumento sem qualquer modificação.

satcomum.br.**as_cnpj**(*numero*)

Formata um número de CNPJ. Se o número não for um CNPJ válido apenas retorna o argumento sem qualquer modificação.

satcomum.br.**as_cnpjcpf**(*numero*)

Formata um número de CNPJ ou CPF. Se o número não for um CNPJ ou CPF válidos apenas retorna o argumento sem qualquer modificação.

satcomum.br.**as_cpf**(*numero*)

Formata um número de CPF. Se o número não for um CPF válido apenas retorna o argumento sem qualquer modificação.

satcomum.br.**cep**(*numero*)

Valida um número de CEP. O número deverá ser informado como uma string contendo 8 dígitos numéricos. Se o número informado for inválido será lançada a exceção `NumeroCEPError`.

> **Aviso:** Qualquer string que contenha 8 dígitos será considerada como um CEP válido, desde que os dígitos não sejam todos iguais.

satcomum.br.**cnpj**(*numero*)
Valida um número de CNPJ. O número deverá ser informado como uma string contendo 14 dígitos numéricos. Se o número informado for inválido será lançada a exceção `NumeroCNPJError`. Esta implementação da validação foi delicadamente copiada de *python-sped <http://git.io/vfuGW>*.

satcomum.br.**cnpjcpf**(*numero*)
Valida um número de CNPJ ou CPF. Veja `cnpj()` e/ou `cpf()`.

satcomum.br.**codigo_ibge_uf**(*sigla*)
Retorna o código do IBGE para a UF informada.

satcomum.br.**cpf**(*numero*)
Valida um número de CPF. O número deverá ser informado como uma string contendo 11 dígitos numéricos. Se o número informado for inválido será lançada a exceção `NumeroCPFError`. Esta implementação da validação foi delicadamente copiada de *python-sped <http://git.io/vfuGW>*.

satcomum.br.**is_cep**(*numero*, *estrito=False*)
Uma versão conveniente para usar em testes condicionais. Apenas retorna verdadeiro ou falso, conforme o argumento é validado.

Parâmetros **estrito** (`bool`) – Padrão `False`, indica se apenas os dígitos do número deverão ser considerados. Se verdadeiro, potenciais caracteres que formam a máscara serão removidos antes da validação ser realizada.

satcomum.br.**is_cnpj**(*numero*, *estrito=False*)
Uma versão conveniente para usar em testes condicionais. Apenas retorna verdadeiro ou falso, conforme o argumento é validado.

Parâmetros **estrito** (`bool`) – Padrão `False`, indica se apenas os dígitos do número deverão ser considerados. Se verdadeiro, potenciais caracteres que formam a máscara serão removidos antes da validação ser realizada.

satcomum.br.**is_cnpjcpf**(*numero*, *estrito=False*)
Uma versão conveniente para usar em testes condicionais. Apenas retorna verdadeiro ou falso, conforme o argumento é validado.

Parâmetros **estrito** (`bool`) – Padrão `False`, indica se apenas os dígitos do número deverão ser considerados. Se verdadeiro, potenciais caracteres que formam a máscara serão removidos antes da validação ser realizada.

satcomum.br.**is_codigo_uf**(*codigo_ibge*)
Indica se o código da UF é um código válido.

satcomum.br.**is_cpf**(*numero*, *estrito=False*)
Uma versão conveniente para usar em testes condicionais. Apenas retorna verdadeiro ou falso, conforme o argumento é validado.

Parâmetros **estrito** (`bool`) – Padrão `False`, indica se apenas os dígitos do número deverão ser considerados. Se verdadeiro, potenciais caracteres que formam a máscara serão removidos antes da validação ser realizada.

satcomum.br.**is_uf**(*sigla*)
Uma versão conveniente para usar em testes condicionais. Apenas retorna verdadeiro ou falso, conforme o argumento é validado.

satcomum.br.**uf**(*sigla*)
Valida a sigla da Unidade Federativa. Se não for uma sigla de UF válida, será lançada a exceção `UnidadeFederativaError`.

satcomum.br.**uf_pelo_codigo**(*codigo_ibge*)
Retorna a UF para o código do IBGE informado.

3.2 Módulo satcomum.constantes

satcomum.constantes.**ASSINATURA_AC_TESTE = 'SGR-SAT SISTEMA DE GESTAO E RETAGUARDA DO SAT'**
Assinatura do Aplicativo Comercial utilizada nos documentos de venda e cancelamento realizados contra equipamentos SAT para desenvolvimento.

satcomum.constantes.**ROOT_TAG_CANCELAMENTO = 'CFeCanc'**
Nome da tag raiz para documentos CF-e-SAT de cancelamento.

satcomum.constantes.**ROOT_TAG_VENDA = 'CFe'**
Nome da tag raiz para documentos CF-e-SAT de venda.

satcomum.constantes.**VERSAO_ER = '2.28.05'**
Versão da Especificação de Requisitos (ER) do SAT-CF-e na qual esta implementação se baseia.

satcomum.constantes.**VERSAO_LAYOUT_ARQUIVO_DADOS_AC = '0.07'**
Versão do layout do arquivo de dados enviado pelo Aplicativo Comercial.

satcomum.constantes.**XML_DECL = '<?xml version="1.0" ?>'**
Declaração de documento XML.

satcomum.constantes.**XML_DECL_UNICODE = '<?xml version="1.0" encoding="utf-8" ?>'**
Declaração de documento XML incluindo a codificação de texto UTF-8.

3.3 Módulo satcomum.ersat

class satcomum.ersat.**ChaveCFeSAT**(*chave*)
Representa a **chave de acesso** do CF-e-SAT conforme descrito na Especificação de Requisitos SAT, item 4.7. Os campos são definidos assim:

```
0  2    6             20 22        31     37     43  --> índice
|  |    |              |  |         |      |      |
35 1508 08723218000186 59 900004019 000024 111425 7  --> campos
|  |    |              |  |         |      |      |
|  |    |              |  |         |      |      dígito verificador
|  |    |              |  |         |      |
|  |    |              |  |         |      código aleatório
|  |    |              |  |         |
|  |    |              |  |         número do cupom fiscal
|  |    |              |  |
|  |    |              |  número de série do equipamento SAT
|  |    |              |
|  |    |              modelo do documento fiscal
|  |    |
|  |    cnpj do emitente
|  |
|  ano/mês de emissão
|
código da UF
```

partes(*num_partes=11*)
Particiona a chave do CF-e-SAT em uma lista de *n* segmentos.

Parâmetros **num_partes** (*int*) – O número de segmentos (partes) em que os digitos da chave do CF-e-SAT serão particionados. **Esse número deverá resultar em uma divisão inteira por 44 (o comprimento da chave)**. Se não for informado, assume `11` partes, comumente utilizado para apresentar a chave do CF-e-SAT no extrato.

Retorna Lista de strings contendo a chave do CF-e-SAT particionada.

Tipo de retorno list

`satcomum.ersat.`**`dados_qrcode`**(*cfe*)

Compila os dados que compõem o QRCode do CF-e-SAT, conforme a documentação técnica oficial **Guia para Geração do QRCode pelo Aplicativo Comercial**, a partir de uma instância de `ElementTree` que represente a árvore do XML do CF-e-SAT.

Parâmetros **`cfe`** – Instância de `xml.etree.ElementTree.ElementTree`.

Retorna String contendo a massa de dados para ser usada ao gerar o QRCode.

Tipo de retorno str

Por exemplo, para gerar a imagem do QRCode[1]:

```
import xml.etree.ElementTree as ET
import qrcode

with open('CFe_1.xml', 'r') as fp:
    tree = ET.parse(fp)
    imagem = qrcode.make(dados_qrcode(tree))
```

`satcomum.ersat.`**`meio_pagamento`**(*codigo*)

Obtém a descrição para o código do meio de pagamento.

Parâmetros **`codigo`** – Código do meio de pagamento, conforme elemento WA03 `cMP`.

Retorna Descrição para o código do meio de pagamento.

Tipo de retorno unicode

3.4 Módulo `satcomum.util`

`satcomum.util.`**`digitos`**(*valor*)

Resulta em uma string contendo apenas os dígitos da string original.

`satcomum.util.`**`modulo11`**(*base*)

Calcula o dígito verificador (DV) para o argumento usando "Módulo 11".

Parâmetros **`base`** (*`str`*) – String contendo os dígitos sobre os quais o DV será calculado, assumindo que o DV não está incluído no argumento.

Retorna O dígito verificador calculado.

Tipo de retorno int

`satcomum.util.`**`texto_decimal`**(*valor*, *remover_zeros=True*)

Converte um valor `decimal.Decimal` para texto, com a opção de remover os zeros à direita não significativos. A conversão para texto irá considerar o `locale` para converter o texto pronto para apresentação.

Parâmetros

- **`valor`** (*`decimal.Decimal`*) – Valor a converter para texto.
- **`remover_zeros`** (*`bool`*) – *Opcional* Indica se os zeros à direita não significativos devem ser removidos do texto, o que irá incluir o separador decimal se for o caso.

[1] https://pypi.python.org/pypi/qrcode

`satcomum.util.`**`validar_casas_decimais`**(*valor*, *minimo=1*, *maximo=2*)

Valida o número de casas decimais. Se o número de casas decimais não estiver dentro do mínimo e máximo, será lançada uma exceção do tipo `ValueError`.

Parâmetros

- **`valor`** – Um objeto `Decimal`.
- **`minimo`** – Valor inteiro maior ou igual a zero indicando o número mínimo de casas decimais. Se não informado, `1` é o mínimo.
- **`maximo`** – Valor inteiro maior ou igual a zero indicando o número máximo de casas decimais. Se não informado, `2` é o máximo.

Levanta `ValueError` – Se o valor possuir um número de casas decimais fora dos limites mínimo e máximo informados.

CAPÍTULO 4

Tabelas e Índices

- genindex
- modindex
- search

Índice de Módulos Python

S

satcomum.br, 7
satcomum.constantes, 9
satcomum.ersat, 9
satcomum.util, 10

Índice

A

as_cep() (*no módulo satcomum.br*), 7
as_cnpj() (*no módulo satcomum.br*), 7
as_cnpjcpf() (*no módulo satcomum.br*), 7
as_cpf() (*no módulo satcomum.br*), 7
ASSINATURA_AC_TESTE (*no módulo satcomum.constantes*), 9

C

cep() (*no módulo satcomum.br*), 7
ChaveCFeSAT (*classe em satcomum.ersat*), 9
cnpj() (*no módulo satcomum.br*), 7
cnpjcpf() (*no módulo satcomum.br*), 8
codigo_ibge_uf() (*no módulo satcomum.br*), 8
cpf() (*no módulo satcomum.br*), 8

D

dados_qrcode() (*no módulo satcomum.ersat*), 10
digitos() (*no módulo satcomum.util*), 10

I

is_cep() (*no módulo satcomum.br*), 8
is_cnpj() (*no módulo satcomum.br*), 8
is_cnpjcpf() (*no módulo satcomum.br*), 8
is_codigo_uf() (*no módulo satcomum.br*), 8
is_cpf() (*no módulo satcomum.br*), 8
is_uf() (*no módulo satcomum.br*), 8

M

meio_pagamento() (*no módulo satcomum.ersat*), 10
modulo11() (*no módulo satcomum.util*), 10

N

NumeroCEPError, 7
NumeroCNPJCPFError, 7
NumeroCNPJError, 7
NumeroCPFError, 7

P

partes() (*método satcomum.ersat.ChaveCFeSAT*), 9

R

ROOT_TAG_CANCELAMENTO (*no módulo satcomum.constantes*), 9
ROOT_TAG_VENDA (*no módulo satcomum.constantes*), 9

S

satcomum.br (*módulo*), 7
satcomum.constantes (*módulo*), 9
satcomum.ersat (*módulo*), 9
satcomum.util (*módulo*), 10

T

texto_decimal() (*no módulo satcomum.util*), 10

U

uf() (*no módulo satcomum.br*), 8
uf_pelo_codigo() (*no módulo satcomum.br*), 8
UnidadeFederativaError, 7

V

validar_casas_decimais() (*no módulo satcomum.util*), 10
VERSAO_ER (*no módulo satcomum.constantes*), 9
VERSAO_LAYOUT_ARQUIVO_DADOS_AC (*no módulo satcomum.constantes*), 9

X

XML_DECL (*no módulo satcomum.constantes*), 9
XML_DECL_UNICODE (*no módulo satcomum.constantes*), 9