python-transip

Release 0.6.0

Roald Nefs

Jan 03, 2023

CONTENTS

python-transip is an Python wrapper for the **TransIP REST API V6**.

CHAPTER

ONE

INTRODUCTION

Welcome to the Python TransIP documentation.

REST API V6

python-transip implements the new TransIP REST API to manage all products in a RESTful way, this means that for most products you can make simple create, update and delete calls.

The wrapper tries to stay close to the naming convention of the official API documentation.

SOAP V5 API (deprecated)

Users are strongly advised to use the new REST API because the SOAP API has been deprecated. If you still depend on functionality that has not yet been implemented in **python-transip** consider raising a feature request and using transip-api (*deprecated*) in the meanwhile.

1.1 Installation

python-transip is available on PyPI:

```
$ python -m pip install python-transip
```

python-transip officially supports Python 3.6+.

1.2 Documentation

The full API Reference and User Guide is available on Read the Docs.

1.3 Authentication

In order to manage TransIP products via **python-transip** you will need to be authenticated. The **REST API V6** requires an access token that makes use of the JSON Web Token standard.

To get an access token, you should first generate a key pair using the control panel. You can than pass the private key to the **python-transip** client to allow the client to generate a new access token, e.g.:

```
import transip

# You can initialize a TransIP client using a private key directly.
PRIVATE_KEY = '''-----BEGIN PRIVATE KEY-----
```

(continues on next page)

(continued from previous page)

```
...
'''
client = transip.TransIP(login='demouser', private_key=PRIVATE_KEY)

# You can also initialize a TransIP client by telling it where to find the
# private key file on the system.
client = transip.TransIP(login='demouser', private_key_file='/path/to/private.key')
```

Alternatively you can also authenticate by providing an access token. This is especially useful when testing the API with the demo token, e.g.:

```
import transip
from transip.v6 import DEMO_TOKEN

# You can initialize a TransIP client using an access token directly.
client = transip.TransIP(access_token=DEMO_TOKEN)
```

CHAPTER

TWO

GENERAL

The general TransIP API resources allow you to manage products, availability zones and call the API test resource.

2.1 Products

Manage available TransIP products and product specifications.

2.1.1 The Product class

When listing all products available on TransIP, a list of **transip.v6.objects.Product** objects is returned.

***class* Product**

The **Product** class makes the following attributes available:

- **name**: The name of the product.
- **description**: The product description.
- **price**: The price in cents.
- **recurringPrice**: The recurring price for the product in cents.
- **elements**: The service to list detailed information on the product elements.

2.1.2 The ProductElement class

When listing all product elements of a **transip.v6.objects.Product** object, a list of **transip.v6.objects.ProductElement** objects is returned.

***class* ProductElement**

The **ProductElement** class makes the following attributes available:

- **name**: The name of the product element.
- **description**: The product element description.
- **amount**: The amount.

2.1.3 List all products

Retrieve al list of products with, there name, description and price.

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# List all available TransIP products.
products = client.products.list()
# Show product information on the screen.
for product in products:
    print((
        f"Product {product.name} ({product.description}) costs "
        f"{product.price} (cents), after which it costs "
        f"{product.recurringPrice} (cents)"
    ))
```

2.1.4 List specifications for product

Get the specification for a product. This will list all the different elements for a product with the amount that it comes with, e.g. a a `vps-bladevps-x4` has 2 CPU-core elements.

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# Show product specification information on the screen for all available
# TransIP products.
for product in client.products.list():
    print(f"Product {product.name} ({product.description}):")
    elements = product.elements.list()
    for element in elements:
        print(f"- Has {element.amount} {element.name}: {element.description}")
```

2.2 Availability Zones

Manage TransIP availability zones.

2.2.1 The AvailabilityZone class

When listing all the available availability zones on TransIP, a list of **transip.v6.objects.AvailabilityZone** objects is returned.

***class* AvailabilityZone**

The **AvailabilityZone** class makes the following attributes available:

- **name**: The name of the availability zone.
- **country**: The 2 letter code for the country the AvailabilityZone is in.
- **isDefault**: If true this is the default zone new VPSes and clones are created in

2.2.2 List availability zones

Retrieve the available availability zones:

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# List all available availability zones.
zones = client.availability_zones.list()
# Show availability zone information on the screen.
for zone in zones:
    print((
        f"Availability zone {zone.name} in {zone.country} "
        f"(default: {zone.isDefault})"
    ))
```

2.3 API Test

A simple test resource to make sure everything is working.

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# Check if everything is working by calling the API test resource.
if client.api_test.test():
    print("Everthing is working!")
```

CHAPTER

THREE

ACCOUNT

The account TransIP API resources allow you to manage invoices and SSH keys.

3.1 Invoices

Manage invoices attached to your TransIP account.

3.1.1 The Invoice class

When listing all invoices attached to your TransIP account, a list of **transip.v6.objects.Invoice** objects is returned.

***class* Invoice**

The **Invoice** class makes the following attributes available:

- **invoiceNumber**: The invoice number.
- **creationDate**: The invoice creation date.
- **payDate**: The invoice paid date.
- **dueDate**: The invoice deadline.
- **invoiceStatus**: The invoice status.
- **currency**: The currency used for this invoice.
- **totalAmount**: The invoice total (*displayed in cents*).
- **totalAmountInclVat**: The invoice total including VAT (*displayed in cents*).
- **items**: The service to list detailed information on the individual invoice items.

3.1.2 The InvoiceItem class

When listing all invoices items attached to a **transip.v6.objects.Invoice** object, a list of **transip.v6.objects.InvoiceItem** objects is returned.

***class* InvoiceItem**

The **InvoiceItem** class makes the following attributes available:

- **product**: The product name.
- **description**: The product description.
- **isRecurring**: Whether or not the payment is recurring.

- **date**: The date when the order line item was up for invoicing.
- **quantity**: The quantity of the invoice item.
- **price**: The price excluding VAT (*displayed in cents*).
- **priceInclVat**: The price including VAT (*displayed in cents*).
- **vat**: The amount of VAT charged.
- **vatPercentage**: The percentage used to calculate the VAT.
- **discounts**: The dictionary containing the applied discounts.
  - **description**: The applied discount description.
  - **amount**: The discounted amount (*in cents*).

The class has the following methods:

- **pdf(_file*path*)** stores the invoice as a PDF file on a location provided by the **file_path** keyword argument. When the **file_path** is a directory, the PDF file will be saved using the invoice number as its basename.

3.1.3 List all invoices

Retrieve all invoices attached to your TransIP account by calling **transip.TransIP.invoices.list()**. This will return a list of **transip.v6.objects.Invoice** objects.

For example:

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# List all invoices attached to your TransIP account.
invoices = client.invoices.list()
# Show invoice information on the screen.
for invoice in invoices:
    print(f"Invoice {invoice.invoiceNumber} was paid on {invoice.payDate}")
```

Note: when using the demo access token, the API currently doesn't list any invoices.

3.1.4 List a single invoice

Retrieve a single invoice attached to your TransIP account by its invoice number by calling **transip.TransIP.invoices.get(*id*)**. This will return a **transip.v6.objects.Invoice** object.

For example:

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# Retrieve a single invoice by its invoice number.
invoice = client.invoices.get('F0000.1911.0000.0004')
# Show invoice information on the screen.
print(f"Invoice {invoice.invoiceNumber} was paid on {invoice.payDate}")
```

Note: when using the demo access token, the API currently doesn't list any invoices.

3.1.5 List invoice items by invoice number

Retrieve the invoice items of a single invoice attached to your TransIP account by its invoice number by calling **items.list()** on a **transip.v6.objects.Invoice** object. This will return a list of **transip.v6.objects.InvoiceItem** objects.

For example:

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# Retrieve a single invoice by its invoice number.
invoice = client.invoices.get('F0000.1911.0000.0004')
# Retrieve all items of a single invoice.
items = invoice.items.list()
# Show invoice information on the screen.
for item in items:
    print(f"Product {item.product} ({item.description})")
```

Note: when using the demo access token, the API currently doesn't list any invoices.

3.1.6 Retrieve an invoice as PDF file

Any of the invoices can be saved as a PDF file by calling **pdf(_file*path*)** on a **transip.v6.objects.Invoice** object:

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# Retrieve a single invoice by its invoice number.
invoice = client.invoices.get('F0000.1911.0000.0004')
# Save the invoice as a PDF file.
invoice.pdf('/path/to/invoices/')
```

Note: when using the demo access token, the API currently doesn't list any invoices.

3.2 SSH Keys

3.3 The SshKey class

When listing all SSH keys attached to your TransIP account, a list of **transip.v6.objects.SshKey** objects is returned.

***class* SshKey**

The **SshKey** class makes the following attributes available:

- **id**: The SSH key identifier.
- **key**: The SSH key.
- **description**: The SSH key description (*max 255 chars*).
- **creationDate**: The date when this SSH key was added (*timezone: Europe/Amsterdam*)
- **fingerprint**: MD5 fingerprint of SSH key.

The class has the following methods:

- **delete**() will delete the SSH key in your TransIP account.
- **update**() will send the updated attributes to the TransIP API.

3.3.1 List all SSH keys

Retrieve all SSH keys attached to your TransIP account by calling **transip.TransIP.ssh_keys.list**(). This will return a list of **transip.v6.objects.SshKey** objects.

For example:

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# List all SSH keys attached to your TransIP account.
ssh_keys = client.ssh_keys.list()
# Show SSH key information on the screen.
for ssh_key in ssh_keys:
    print(f"SSH key {ssh_key.id} has fingerprint {ssh_key.fingerprint}")
```

Note: when using the demo access token, the API currently doesn't list any SSH keys.

3.3.2 Get SSH key by id

Retrieve a single SSH key attached to your TransIP account by its ID by calling **transip.TransIP.ssh_key.get**(***id***). This will return a **transip.v6.objects.SshKey** object.

For example:

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# Retrieve a SSH key by its ID (provided by TransIP).
invoice = client.ssh_keys.get(123)
# Show SSH key information on the screen.
print(f"SSH key {ssh_key.id} has fingerprint {ssh_key.fingerprint}")
```

Note: when using the demo access token, the API currently doesn't list any SSH keys.

3.3.3 Add a new SSH key

Add a new SSH key to your TransIP account by calling **transip.TransIP.ssh_key.create**(***data***). The **data** keyword argument requires a dictionary with the **sshKey** and **description** attributes.

For example:

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

```

(continues on next page)

(continued from previous page)

```
# Data used to create a new SSH key.
key_data = {
    "sshKey": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDf2pxWX/
↪yhUBDyk2LPhvRtI0LnVO8PyR5Zt6AHrnhtLGqK+8YG9EMlWbCCWrASR+Q1hFQG example",
    "description": "Jim key"
}
# Add the new SSH key to your TransIP account.
client.ssh_keys.create(key_data)
```

Note: when using the demo access token, the API currently doesn't list any SSH keys.

3.3.4 Update an SSH key

Update an existing SSH key in your TransIP account by calling **transip.TransIP.ssh_key.update(*id*, *data*)**. The **id** keyword argument is the ID of the SSH key provided by TransIP and **data** keyword argument requires a dictionary with the **sshKey** and **description** attributes.

For example:

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# Dictionary containing the new description of the SSH key
key_data = {
    "description": "Jim key"
}
# Update SSH key (ID: 123) with the new description.
client.ssh_keys.update(123, key_data)
```

The **transip.v6.objects.SshKey** class also provides a **update()** method to update a **SshKey** object from an instance after changing any of the update-able attributes.

Note: when using the demo access token, the API currently doesn't list any SSH keys.

3.3.5 Delete an SSH key

Delete an existing SSH key in your TransIP account by calling **transip.TransIP.ssh_key.delete(*id*)**. The **id** keyword argument is the ID of the SSH key provided by TransIP.

For example:

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# Delete SSH key with ID 123.
client.ssh_keys.delete(123)
```

The **transip.v6.objects.SshKey** class also provides a **delete()** method to delete a **SshKey** object from an instance.

Note: when using the demo access token, the API currently doesn't list any SSH keys.

CHAPTER

FOUR

DOMAIN

The domains TransIP API resources allow you to manage domains, branding, contacts, DNS, DNSSEC, nameservers, actions, SSL, WHOIS, availability and call the tlds resource.

The documentation for managing **domains** and related resources has not yet completely been documented. Feel free to file an issue for adding the missing section(s) in the documentation.

4.1 Domains

Manage domains.

4.1.1 The Domain class

When listing all domains in your TransIP account, a list of **transip.v6.objects.Domain** objects is returned.

***class* Domain**

The **Domain** class makes the following attributes available:

- **name**: The name, including the tld of this domain.
- **authCode**: The authcode for this domain as generated by the registry.
- **isTransferLocked**: If this domain supports transfer locking, this flag is True when the domains ability to transfer is locked at the registry.
- **registrationDate**: The registration date of the domain, in YYYY-mm-dd format.
- **renewalDate**: The next renewal date of the domain, in YYYY-mm-dd format.
- **isWhitelabel**: If this domain is added to your whitelabel.
- **cancellationDate**: Cancellation data, in YYYY-mm-dd h:i:s format, None if the domain is active.
- **cancellationStatus**: Cancellation status, None if the domain is active, 'cancelled' when the domain is cancelled.
- **isDnsOnly**: Whether this domain is DNS only.
- **tags**: The custom tags added to this domain.
- **contacts**: The service to manage the domain contacts.
- **dns**: The service to manage the DNS-records of the domain.
- **nameservers**: The service to manage the nameservers of the domain.

4.1.2 List all domains

Retrieve all domains registered in your TransIP account by calling **transip.TransIP.domains.list()**. This will return a list of **transip.v6.objects.Domain** objects.

For example:

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# List all domains.
domains = client.domains.list()
# Show domain information on the screen.
for domain in domains:
    print(f"Domain {domain.name} was registered at {domain.registrationDate}")
```

4.1.3 Retrieve an existing domain

Retrieve a single domain registered ion your TransIP account by its ID by calling **transip.TransIP.domains.get(*name*)**. This will return a **transip.v6.objects.Domain** object.

For example:

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# Retrieve a domain by its name
domain = client.domains.get('transipdemonstratie.nl')
# Show domain information on the screen.
print(f"Domain {domain.name} was registered at {domain.registrationDate}")
```

4.2 DNS

Manage DNS records of a domain. Any changes made here will be pushed to the TransIP nameservers.

4.2.1 The DnsEntry class

When listing all DNS-records of a **transip.v6.objects.Domain** object, a list of **transip.v6.objects.DnsEntry** objects is returned.

***class* DnsEntry**

The **DnsEntry** class makes the following attributes available:

- **name**: The name of the dns entry, for example '@' or 'www'
- **expire**: The expiration period of the dns entry, in seconds. For example 86400 for a day of expiration.
- **type**: The type of dns entry. Possbible types are 'A', 'AAAA', 'CNAME', 'MX', 'NS', 'TXT', 'SRV', 'SSHFP' and 'TLSA'.
- **content**: The content of of the dns entry, for example '10 mail', '127.0.0.1' or 'www'.

The class has the following methods:

- **delete()** will delete the DNS-record from the domain.
- **update()** will send the updated attributes to the TransIP API. This can only be used when updating the **content** attribute of a DnsEntry and when there aren't any other DNS records with the same **name**, **expire** and **type** attributes.

4.2.2 List all DNS entries for a domain

Retrieve the DNS records of a single domain registered in your TransIP account by calling **dns.list()** on a **transip.v6.objects.Domain** object. This will return a list of **transip.v6.objects.DnsEntry** objects.

For example:

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# Retrieve a domain by its name.
domain = client.domains.get('transipdemonstratie.nl')
# Retrieve the DNS records of a single domain.
records = domain.dns.list()
# Show the DNS record information on the screen.
for record in records:
    print(f"DNS: {record.name} {record.expire} {record.type} {record.content}")
```

4.2.3 Add a new single DNS entry to a domain

Add an new DNS record to a domain by calling **dns.create(*data*)** on a **transip.v6.objects.Domain** object. The **data** keyword argument a dictionary containing the **name**, **expire**, **type** and **content** attributes.

For example:

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# Retrieve a domain by its name.
domain = client.domains.get('transipdemonstratie.nl')
# Dictionary containing the information for a single DNS record.
dns_entry_data = {
    "name": "www",
    "expire": 86400,
    "type": "A",
    "content": "127.0.0.1"
}
# Add the DNS record to the domain.
domain.dns.create(dns_entry_data)
```

4.2.4 Update single DNS entry

Update a single DNS record of a domain by calling **dns.update(*data*)** on a **transip.v6.objects.Domain** object. The **data** keyword argument a dictionary containing the **name**, **expire**, **type** and **content** attributes.

This can only be used when updating the **content** attribute of a DNS entry and when there aren't any other DNS records with the same **name**, **expire** and **type** attributes.

For example:

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# Retrieve a domain by its name.
domain = client.domains.get('transipdemonstratie.nl')
# Dictionary containing the information for a single updated DNS record.
dns_entry_data = {
    "name": "www",
    "expire": 86400,
    "type": "A",
    "content": "127.0.0.2"  # The update content.
}
# Update the content of a single DNS record.
domain.dns.update(dns_entry_data)
```

4.2.5 Update all DNS entries for a domain

Update all DNS records of a single domain registered in your TransIP account at once by calling **dns.replace()** on a **transip.v6.objects.Domain** object.

Note: This will wipe all existing DNS records with the provided records.

For example:

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# Retrieve a domain by its name.
domain = client.domains.get('transipdemonstratie.nl')
# Retrieve the DNS records of a single domain.
records = domain.dns.list()

for record in records:
    # Update the A-record for localhost
    if record.name == 'localhost' and record.type == 'A':
        record.content = '127.0.0.1'

# Replace all the records with the updated ones
domain.dns.replace(records)
```

4.2.6 Remove a DNS entry from a domain

Delete an existing DNS record from a domain by calling **dns.delete(*data*)** on a **transip.v6.objects.Domain** object. The **data** keyword argument a dictionary containing the **name**, **expire**, **type** and **content** attributes.

For example:

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# Retrieve a domain by its name.
domain = client.domains.get('transipdemonstratie.nl')
# Dictionary containing the information for a single DNS record.
dns_entry_data = {
    "name": "www",
    "expire": 86400,
    "type": "A",
    "content": "127.0.0.1"
}
# Delete the DNS record from the domain.
domain.dns.delete(dns_entry_data)
```

The **transip.v6.objects.DnsEntry** class also provides a **delete()** method to delete a **DnsEntry** object from an instance.

4.3 Nameservers

Manage the nameserver of a domain.

4.3.1 The Nameserver class

When listing all nameservers of a **transip.v6.objects.Domain** object, a list of **transip.v6.objects.Nameserver** objects is returned.

***class* Nameserver**

The **Nameserver** class makes the following attributes available:

- **hostname**: The hostname of this nameserver.
- **ipv4**: The optional ipv4 glue record for this nameserver.
- **ipv6**: The optional ipv6 glue record for this nameserver

4.3.2 List nameservers for a domain

Retrieve the nameserver of a single domain registered in your TransIP account by calling **nameserver.list()** on a **transip.v6.objects.Domain** object. This will return a list of **transip.v6.objects.Nameserver** objects.

For example:

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# Retrieve a domain by its name.
domain = client.domains.get('transipdemonstratie.nl')
# Retrieve the nameservers of a single domain.
nameservers = domain.nameservers.list()
# Show the nameserver information on the screen.
for nameserver in nameservers:
    print(f"Nameserver: {nameserver.hostname}")
```

4.3.3 Update nameservers for a domain

Update all the nameservers of a single domain registered in your TransIP account at once by calling **nameservers.replace()** on a **transip.v6.objects.Domain** object.

Note: This will wipe all nameservers previously configured on the domain.

For example:

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# Retrieve a domain by its name.
domain = client.domains.get('transipdemonstratie.nl')
# Retrieve the nameservers of the domain.
nameservers = domain.nameservers.list()

# Update the ipv4 glue record of the first nameserver.
nameserver[0].ipv4 = '195.135.195.195'

# Replace all the records with the updated ones
domain.nameservers.replace(nameservers)
```

CHAPTER

FIVE

VPS

The documentation for managing **VPSs** and related resources has not yet been documented. Feel free to file an issue for adding the missing section(s) in the documentation.

CHAPTER

SIX

HA-IP

The documentation for managing **HA-IPs** and related resources has not yet been documented. Feel free to file an issue for adding the missing section(s) in the documentation.

CHAPTER

SEVEN

COLOCATION

The documentation for managing **colocations** and related resources has not yet been documented. Feel free to file an issue for adding the missing section(s) in the documentation.

7.1 Colocations

Manage colocations.

7.1.1 The Colocation class

When listing all colocations in your TransIP account, a list of **transip.v6.objects.Colocation** objects is returned.

***class* Colocation**

The **Colocation** class makes the following attributes available:

- **name**: The colocation name.
- **ipRanges**: The list of IP ranges.

7.1.2 List all colocations

Retrieve the colocations registered in your TransIP account by calling **transip.TransIP.colocations.list()**. This will return a list of **transip.v6.objects.Colocation** objects.

For example:

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# Retrieve the colocations registered in your TransIP account.
colocations = client.colocations.list()
# Show the colocations information on the screen.
for colocation in colocations:
    ip_ranges = ' '.join(colocation.ipRanges)
    print(f"Colocation: {colocation.name} has IP ranges: {ip_ranges}")
```

7.1.3 Get colocation

Retrieve a single colocation registered in your TransIP account by its ID by calling **transip.TransIP.colocations.get(*name*)**. This will return a **transip.v6.objects.Colocation** object.

For example:

```
import transip
# Initialize a client using the TransIP demo token.
client = transip.TransIP(access_token=transip.v6.DEMO_TOKEN)

# Retrieve a colocation by its name.
colocation = client.colocations.get('example2')
# Show colocation information on the screen.
ip_ranges = ' '.join(colocation.ipRanges)
print(f"Colocation: {colocation.name} has IP ranges: {ip_ranges}")
```

CHAPTER

EIGHT

FURTHER READING

8.1 Changelog

All notable changes in **python-transip** are documented below.

8.1.1 Unreleased

8.1.2 0.6.0 (2021-11-01)

Added

- Python 3.10 support (#49).

8.1.3 0.5.0 (2021-02-10)

Added

- The option to replace all existing nameservers of a single domain at once from the `transip.v6.objects.Domain.nameservers` service.
- The option to list all colocations from the `transip.TransIP.colocations` service (#24).
- The option to retrieve a single colocation by name from the `transip.TransIP.colocations` service (#24).
- The option to allow the access token to be used from all IP-addresses instead of only the whitelisted ones (#46).

8.1.4 0.4.0 (2021-01-24)

Added

- This `CHANGELOG.md` file to be able to list all notable changes for each version of **python-transip**.
- The `transip.TransIP.api_test` service to allow calling the test resource to make sure everything is working.
- The option to list all invoices attached to your TransIP account from the `transip.TransIP.invoices` service.
- The option to save an invoice as PDF file from `transip.v6.objects.Invoice` object.
- The option to list all products available in TransIP from the `transip.TransIP.products` service.
- The option to update a single SSH key from `transip.v6.objects.SshKey` object.

- The option to update the content of a single DNS record from the `transip.v6.objects.Domain.dns` service, as well as from the `transip.v6.objects.DnsEntry` object.
- The option to replace all existing DNS records of a single domain at once from the `transip.v6.objects.Domain.dns` service.