Connect SDK Documentation

Ingram Micro

Jun 12, 2019

Contents

CHAPTER 1

Introduction

Connect Python SDK allows an easy and fast integration with Connect fulfillment API. Thanks to it you can automate the fulfillment of orders generated by your products.

In order to use this library, please ensure that you have read first the documentation available on Connect knowledge base article located here, this one will provide you a great information on the REST API that this library implements.

This library may be consumed in your project in order to automate the fulfillment of requests, this class once imported into your project will allow you to:

- Communicate with Connect using your API credentials.
- **List all requests, and even filter them:**
  - For a specific product.
  - For a specific status.
  - For a specific asset.
  - Etc.
- Process each request and obtain full details of the request.
- **Modify the activation parameters of each request in order to:**
  - Inquiry for changes
  - Store information into the fulfillment request
- Change the status of the requests from its initial pending state to either inquiring, failed or approved.
- Generate and upload usage files to report usage for active contracts and listings.
- Process usage file status changes.
- Work with Notes for requests.
- Generate logs.
- Collect debug logs in case of failure.

Your code may use any scheduler to execute, from a simple cron to a cloud scheduler like the ones available in Azure, Google, Amazon or other cloud platforms.

1.1 Installation

In order to use the SDK with your product, you must install the `connect-sdk` package from PyPI (Python Package Index). You can do so using `pip`:

```
$ pip install connect-sdk
```

1.2 Requirements

- Python 2.7+ or Python 3.4+
- Requests
- Marshmallow

CHAPTER 2

Contents

2.1 Examples

2.1.1 Processing Fulfillment Requests

```
# -*- coding: utf-8 -*-

# This file is part of the Ingram Micro Cloud Blue Connect SDK.
# Copyright (c) 2019 Ingram Micro. All Rights Reserved.

from typing import Union
import warnings

from connect.config import Config
from connect.exceptions import FailRequest, InquireRequest, SkipRequest
from connect.logger import logger
from connect.models import ActivationTemplateResponse, ActivationTileResponse,␣
↪Fulfillment
from connect.resources import FulfillmentAutomation

# Enable processing of deprecation warnings
warnings.simplefilter('default')

# Set logger level / default level ERROR
logger.setLevel('DEBUG')

# If we remove this line, it is done implicitly
Config(file='config.json')


class FulfillmentExample(FulfillmentAutomation):
    def process_request(self, request):
        # type: (Fulfillment) -> Union[ActivationTemplateResponse,␣
↪ActivationTileResponse]
```

(continues on next page)

(continued from previous page)

```
        if request.needs_migration():
            # Skip request if it needs migration (migration is performed by an␣
↪external service)
            logger.info('Skipping request {} because it needs migration.'.
↪format(request.id))
            raise SkipRequest()
        else:
            logger.info('Processing request {} for contract {}, product {},␣
↪marketplace {}'
                        .format(request.id,
                                request.contract.id,
                                request.asset.product.name,
                                request.marketplace.name))

            # Custom logic
            if request.type == 'purchase':
                for item in request.asset.items:
                    if item.quantity > 100000:
                        raise FailRequest(
                            message='Is Not possible to purchase product')

                for param in request.asset.params:
                    if param.name == 'email' and not param.value:
                        param.value_error = 'Email address has not been provided, ' \
                                            'please provide one'
                        raise InquireRequest(params=[param])

                # Approve by ActivationTile
                return ActivationTileResponse('\n  # Welcome to Fallball!\n\nYes, you␣
↪decided '
                                              'to have an account in our amazing␣
↪service!')
                # Or
                # return TemplateResource().render(pk='TEMPLATE_ID', request_
↪id=request.id)

                # Approve by Template
                # return ActivationTemplateResponse('TL-497-535-242')
                # Or
                # return TemplateResource().get(pk='TEMPLATE_ID')

            elif request.type == 'change':
                # Fail
                raise FailRequest()
            else:
                # Skip request
                raise SkipRequest()


if __name__ == '__main__':
    fulfillment_example = FulfillmentExample()
    fulfillment_example.process()
```

2.1.2 Processing Tier Config Requests

```
# -*- coding: utf-8 -*-

# This file is part of the Ingram Micro Cloud Blue Connect SDK.
# Copyright (c) 2019 Ingram Micro. All Rights Reserved.

# NOTE: This example development is in progress. This is just a skeleton.

from typing import Union
import warnings

from connect.config import Config
from connect.logger import logger
from connect.models import ActivationTemplateResponse, ActivationTileResponse,␣
↪TierConfigRequest
from connect.resources import TierConfigAutomation

# Enable processing of deprecation warnings
warnings.simplefilter('default')

# Set logger level / default level ERROR
logger.setLevel('DEBUG')

# If we remove this line, it is done implicitly
Config(file='config.json')


class TierConfigExample(TierConfigAutomation):
    def process_request(self, request):
        # type: (TierConfigRequest) -> Union[ActivationTemplateResponse,␣
↪ActivationTileResponse]
        pass


if __name__ == '__main__':
    tier_config_example = TierConfigExample()
    tier_config_example.process()
```

2.1.3 Reporting Usage Files

```
# -*- coding: utf-8 -*-

# This file is part of the Ingram Micro Cloud Blue Connect SDK.
# Copyright (c) 2019 Ingram Micro. All Rights Reserved.

from datetime import date, timedelta
import time
import warnings

from connect.config import Config
from connect.logger import logger
from connect.models import Contract, UsageRecord, UsageFile, UsageListing, Product
from connect.resources import UsageAutomation
```

(continues on next page)

(continued from previous page)

```
# Enable processing of deprecation warnings
warnings.simplefilter('default')

# Set logger level / default level ERROR
logger.setLevel('DEBUG')

# If we remove this line, it is done implicitly
Config(file='config.json')


class UsageExample(UsageAutomation):
    def process_request(self, request):
        # type: (UsageListing) -> None

        # Detect specific provider contract
        if request.contract.id == 'CRD-41560-05399-123':
            # Can also be seen from request.provider.id and parametrized further
            # via marketplace available at request.marketplace.id
            usage_file = UsageFile(
                name='sdk test',
                product=Product(id=request.product.id),
                contract=Contract(id=request.contract.id)
            )

            usages = [
                UsageRecord(
                    record_id='unique record value',

                    item_search_criteria='item.mpn',
                    # Possible values are item.mpn or item.local_id.

                    item_search_value='SKUA',
                    # Value defined as MPN on vendor portal.

                    quantity=1,
                    # Quantity to be reported.

                    start_time_utc=(date.today() - timedelta(1)).strftime('%Y-%m-%d'),
                    # From when to report.

                    end_time_utc=time.strftime('%Y-%m-%d %H:%M:%S'),
                    # Till when to report.

                    asset_search_criteria='parameter.param_b',
                    # How to find the asset on Connect.  Typical use case is to use a␣
↪parameter
                    # provided by vendor, in this case called param_b.  Additionally,␣
↪asset.id
                    # can be used in case you want to use Connect identifiers.

                    asset_search_value='tenant2'
                )
            ]

            self.submit_usage(usage_file, usages)
        else:
            # Something different could be done here
```

(continues on next page)

(continued from previous page)

```
            pass


if __name__ == '__main__':
    usage_example = UsageExample()
    usage_example.process()
```

2.1.4 Workflow of Usage Files

```
# -*- coding: utf-8 -*-

# This file is part of the Ingram Micro Cloud Blue Connect SDK.
# Copyright (c) 2019 Ingram Micro. All Rights Reserved.

import warnings

from connect.config import Config
from connect.logger import logger
from connect.exceptions import AcceptUsageFile, DeleteUsageFile, SkipRequest,␣
↪SubmitUsageFile
from connect.models import UsageFile
from connect.resources import UsageFileAutomation

# Enable processing of deprecation warnings
warnings.simplefilter('default')

# Set logger level / default level ERROR
logger.setLevel('DEBUG')

# If we remove this line, it is done implicitly
Config(file='config.json')


class UsageFileExample(UsageFileAutomation):
    def process_request(self, request):
        # type: (UsageFile) -> None
        if request.status == 'invalid':
            # Vendor and provider may handle invalid cases differently,
            # probably notifying their staff
            raise DeleteUsageFile('Not needed anymore')
        elif request.status == 'ready':
            # Vendor may submit file to provider
            raise SubmitUsageFile('Ready for provider')
        elif request.status == 'pending':
            # Provider use case, needs to be reviewed and accepted
            raise AcceptUsageFile('File looks good')
        else:
            raise SkipRequest('Non controlled status')


if __name__ == '__main__':
    usage_file_example = UsageFileExample()
    usage_file_example.process()
```

2.2 API Reference

2.2.1 config

2.2.2 exceptions

2.2.3 models

2.2.4 resources

CHAPTER 3

Indices and tables

- genindex
- modindex
- search