Connect SDK Documentation

Ingram Micro

Jun 05, 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

class connect.config.**Config**(*api_url=None*, *api_key=None*, *products=None*, *file=None*)

Initialization config for public api.

Parameters

- **api_url** (*str*) – Public api url.
- **api_key** (*str*) – Service user ApiKey.
- **products** (*str | list[str]*) – Optional product ids.
- **file** (*str*) – Config file name.

Raises

- **ValueError** – Raised if either `file` or one of `api_url` or `api_key` are missing.
- **TypeError** – Raised if `products` is not a string or list of strings, or if config file does not contain JSON data.
- **IOError** – Raised if the specified `file` could not be opened.

api_key

Returns ApiKey.

Return type str

api_url

Returns Api URL.

Return type str

classmethod get_instance()

Returns Global instance.

Return type *Config*

products

Returns Valid product ids.

Return type list[str]

2.2.2 exceptions

exception connect.exceptions.**AcceptUsageFile**(*acceptance_note*)

exception connect.exceptions.**CloseUsageFile**(*message=None*)

exception connect.exceptions.**DeleteUsageFile**(*message=None*)

exception connect.exceptions.**FailRequest**(*message=”*)

Causes the request being processed to fail.

Parameters message (*str*) – Exception message.

exception connect.exceptions.**FileCreationError**(*message*)

exception connect.exceptions.**FileRetrievalError**(*message*)

exception `connect.exceptions.`**`InquireRequest`**(*message=”*, *params=None*)

Causes the request being processed to inquire for some information.

Parameters

- **`message`** (*`str`*) – Exception message.
- **`params`** (*`List[Param]`*) – Parameters to inquire.

`params = None`

(List[*`Param`*]) Parameters to inquire.

exception `connect.exceptions.`**`Message`**(*message=”*, *code=”*, *obj=None*)

Base class for all Connect exceptions.

Parameters

- **`message`** (*`str`*) – Exception message.
- **`code`** (*`str`*) – Exception code.
- **`obj`** (*`object`*) – Additional information.

`code = None`

(str) Exception code.

`message`

Returns The exception message.

Return type str

Deprecated since version 16.0: Use `str(exception)` instead.

`obj = None`

(str) Additional information.

exception `connect.exceptions.`**`RejectUsageFile`**(*message=None*)

exception `connect.exceptions.`**`ServerError`**(*error*)

Indicates that the server returned an error.

Parameters **`error`** (`ServerErrorResponse`) – Response returned by the server.

exception `connect.exceptions.`**`SkipRequest`**(*message=”*)

Causes the request being processed to be skipped.

Parameters **`message`** (*`str`*) – Exception message.

exception `connect.exceptions.`**`SubmitUsageFile`**(*rejection_note*)

exception `connect.exceptions.`**`UsageFileAction`**(*message*, *code*, *data=None*)

Base exception for Usage API actions.

Parameters

- **`message`** (*`str`*) – Exception message.
- **`code`** (*`str`*) – Exception code.
- **`data`** (*`Optional[Dict[str,Any]]`*) – Additional information.

2.2.3 models

class `connect.models.`**`Activation`**(***kwargs*)

Activation object.

date = None
(datetime.datetime|None) Activation date.

link = None
(str|None) Activation link.

message = None
(str) Activation message.

class connect.models.**ActivationTemplateResponse**(*template_id*)
An instance of this class might the returned by the overriden `process_request` method of your `connect.resources.FulfillmentAutomation` or `connect.resources.TierConfigAutomation` subclass to approve the request being processed, showing a tile with the specified template id.

Parameters **template_id** (*str*) – Id of the template od the tile to be shown in the Vendor Portal. The template must have been defined in the Vendor Portal.

Return type None

class connect.models.**ActivationTileResponse**(*markdown=''*)
An instance of this class might the returned by the overriden `process_request` method of your `connect.resources.FulfillmentAutomation` or `connect.resources.TierConfigAutomation` subclass to approve the request being processed, showing a tile with the specified contents.

Parameters **markdown** (*str*) – Contents of the tile to be shown in the Vendor Portal, in Markdown format.

Return type None

class connect.models.**Agreement**(***kwargs*)
An Agreement object.

active = None
(bool) State of the version.

agreements = None
(List[`Agreement`]) Program agreements can have distribution agreements associated with them.

author = None
(`User` | None) Reference to the user who created the version.

created = None
(datetime.datetime) Date of creation of the agreement.

description = None
(str) Agreement details (Markdown).

link = None
(str) Url to the document.

marketplace = None
(`Marketplace` | None) Reference to marketplace object (for distribution agreement).

owner = None
(`Company`) Reference to the owner account object.

parent = None
(`Agreement` | None) Reference to the parent program agreement (for distribution agreement).

stats = None
(`AgreementStats` | None) Agreement stats.

title = None
(str) Title of the agreement.

type = None
(str) Type of the agreement. One of: distribution, program, service.

updated = None
(datetime.datetime) Date of the update of the agreement. It can be creation of the new version, change of the field, etc. (any change).

version = None
(int) Chronological number of the version.

version_contracts = None
(int) Number of contracts this version has.

version_created = None
(datetime.datetime) Date of the creation of the version.

class `connect.models.`**AgreementStats**(***kwargs*)
Agreement stats.

contracts = None
(int|None) Number of contracts this agreement has.

versions = None
(int) Number of versions in the agreement.

class `connect.models.`**Asset**(***kwargs*)
Represents a saleable item that can be provided/distributed in terms of one purchase.

These assets can be requested using `connect.resource.FulfillmentAutomation` resource.

An asset is characterized by the following:

- Every asset reflects some purchase (somebody purchases either a service or a good).
- Purchase action can be reverted (canceled) or terminated when terms of purchase are expired, see full state diagram on FIG.5
- Asset can be subscription-based (when customer pay for usage in some time terms) or one-time based.
- Matter of asset is defined as list of purchased items with purchased quantities (asset items).
- Item in asset may be either reservation-based, when customer decides how many items of SKU to be purchased or Pay-Per-User based when actual use of the SKU defines quantity for asset item.
- Asset may be modified using change requests: either set of items may be changed or quantities of reservation-based items may be changed.
- Some assets can be put into suspend state, when service is not actually provided and no charges happened.
- Assets also may be parametrized by one or more parameters which are differentiate one asset from another.

connection = None
(*`Connection`*) Connection object.

external_id = None
(str) Identification for asset object on eCommerce.

external_uid = None
(str|None) Id of asset in eCommerce system

get_item_by_mpn(*mpn*)
Get an item of the asset.

Parameters **mpn** (*`str`*) – MPN of the item to get.

Returns The item with the given MPN, or `None` if it was not found.

Return type `Item` | None

`get_param_by_id`(*id_*)

Get a parameter of the asset.

Parameters **`id`** (`str`) – Id of the the parameter to get.

Returns The parameter with the given id, or `None` if it was not found.

Return type `Param` | None

`items = None`

(List[`Item`]) List of asset product items.

`params = None`

(List[`Param`]) List of product parameter objects.

`product = None`

(`Product`) Product object reference.

`status = None`

Assets may have one of the following statuses:

- new: First purchase requested.
- processing: Until first purchase request is either completed or rejected.
- **active: After the first purchase request is completed.** NOTE: Asset stays active regardless of any other requests except cancel.
- rejected: Asset becomes rejected once the first purchase request is rejected.
- terminated: Asset becomes terminated once the 'cancel' request type is fulfilled.
- suspended: Asset becomes suspended once 'suspend' request type is fulfilled.

`tiers = None`

(`TierAccounts`) Supply chain accounts.

class `connect.models.`**`BaseModel`**(***kwargs*)

Base class of all models.

All the arguments provided on creation of the model are injected as attributes on the object.

classmethod deserialize(*json_str*)

Deserialize a string containing JSON data into a model.

Parameters **`json_str`** (`str`) – String containing the JSON data to be deserialized.

Returns An instance of the same class as the receiver of the call, or a list of instances.

Return type Any|list[Any]

Raises **`TypeError`** – Raised if the data cannot be deserialized.

classmethod deserialize_json(*json_data*)

Deserialize JSON data into a model.

Parameters **`json_data`** (`dict | list`) – JSON list or dictionary to be deserialized.

Returns An instance of the same class as the receiver of the call, or a list of instances.

Return type Any|list[Any]

Raises **`TypeError`** – Raised if the data cannot be deserialized.

`id = None`

(str) Globally unique id.

json

Returns The JSON representation of the model.

Return type dict|list

class connect.models.**Company**(***kwargs*)

Represents a company within the platform.

name = None

(str) Company name.

class connect.models.**Connection**(***kwargs*)

Represents a communication channel which provides the ability to order products within particular hub.

Standalone connection is required for each product and for each provider account.

hub = None

(*Hub*) Hub Reference.

product = None

(*Product*) Product Reference.

provider = None

(*Company*) Provider Account Reference.

type = None

(str) Type of connection.

vendor = None

(*Company*) Vendor Account Reference.

class connect.models.**Constraints**(***kwargs*)

Parameter constraints.

choices = None

(List[*ValueChoice*]) Parameter value choices.

hidden = None

(bool) Is the parameter hidden?

required = None

(bool) Is the parameter required?

class connect.models.**Contact**(***kwargs*)

Person of contact.

email = None

(str) Email address.

first_name = None

(str|None) First name.

last_name = None

(str|None) Last name.

phone_number = None

(*PhoneNumber*) Phone number.

class connect.models.**ContactInfo**(***kwargs*)

Represents the information of a contact.

address_line1 = None

(str) Street address, first line.

address_line2 = None
(str|None) Street address, second line.

city = None
(str) City name.

contact = None
(*Contact*) Person of contact.

country = None
(str) Country code.

postal_code = None
(str) Postal ZIP code.

state = None
(str) State name.

class connect.models.**Contract**(***kwargs*)
Contract object.

activation = None
(*Activation*) Activation information.

agreement = None
(*Agreement*) Reference object to the agreement.

created = None
(datetime.datetime) Contract creation date.

creator = None
(*User*) Reference object to the creator.

enrolled = None
(datetime.datetime|None) Date when contract was enrolled.

marketplace = None
(*Marketplace* | None) Reference object to the agreement marketplace.

name = None
(str) Contract name.

owner = None
(*Company* | None) Reference object to the owner company.

signee = None
(*User* | None) Reference object to the user of the owner company, who signed the contract.

status = None
(str) Contract Status. One of: enrolling, pending, active, terminated, rejected

type = None
(str) Type of the contract (same as agreement type). One of: distribution, program, service.

updated = None
(datetime.datetime) Date of contract status update.

version = None
(int) Version of the contract (same as associated agreement version).

version_created = None
(datetime.datetime) Contract version creation date.

class connect.models.**Conversation**(***kwargs*)

Conversation.

add_message(*message*, *config=None*)

Adds a message to the conversation.

Parameters

- **message** (*str*) – Message to add.
- **config** (Config) – Configuration, or None to use the environment config (default).

Returns The added message.

Return type *ConversationMessage*

Raises **TypeError** – Raised if the message cannot be deserialized.

created = None

(datetime.datetime) Date of the Conversation creation.

creator = None

(*User*) Creator of the conversation.

instance_id = None

(str) The id of object based on which discussion is made, e.g. listing request. It can be any object.

messages = None

(List[*ConversationMessage*]) List of *ConversationMessage* objects.

topic = None

(str) Conversation topic.

class connect.models.**ConversationMessage**(***kwargs*)

Message in a *Conversation*.

conversation = None

(str) Primary ID of Conversation object.

created = None

(datetime.datetime) Date of the Message creation.

creator = None

(*User*) *User* that created the message.

text = None

(str) Actual message.

class connect.models.**CustomerUiSettings**(***kwargs*)

Customer Ui Settings for a product.

description = None

(str) Description.

documents = None

(List[*Document*]) Documents.

download_links = None

(List[*DownloadLink*]) Download links.

getting_started = None

(str) Getting started.

class connect.models.**Document**(***kwargs*)

Document for a product.

title = None
(str) Document title.

url = None
(str) Document URL.

visible_for = None
(str) Document visibility. One of: admin, user.

class connect.models.**DownloadLink**(***kwargs*)
Download link for a product.

title = None
(str) Link title.

url = None
(str) Link URL.

class connect.models.**EventInfo**(***kwargs*)
Represents the date and user that caused an event.

at = None
(datetime.datetime|None) Date when the event occurred.

by = None
(*User*) User that caused the event.

class connect.models.**Events**(***kwargs*)
Represents a set of events that can take place on an object.

created = None
(*EventInfo*) Creation event.

inquired = None
(*EventInfo*) Inquire event.

pended = None
(*EventInfo*) Pending event.

updated = None
(*EventInfo*) Update event.

validated = None
(*EventInfo*) Validation event.

class connect.models.**ExtIdHub**(***kwargs*)
Associates a *Hub* with an external id.

external_id = None
(str) External id.

hub = None
(*Hub*) Hub.

class connect.models.**Fulfillment**(***kwargs*)
Represents a request for the `connect.resource.FulfillmentAutomation` resource.

activation_key = None
(str) Activation key content for activating the subscription on vendor portal. This markdown formatted message is sent to customer.

asset = None
(*Asset*) Asset object.

changed_items

Returns Changed items.

Return type List[*Item*]

contract = None
(`Contract`) Contract object.

created = None
(datetime.datetime) Date of request creation.

get_conversation (*config=None*)

Parameters **config** (`Config`) – Configuration, or `None` to use the environment config (default).

Returns The conversation for this request, or `None` if there is none.

Return type Conversation|None

marketplace = None
(`Marketplace`) Marketplace object.

needs_migration (*migration_key='migration_info'*)
Indicates whether the request contains data to be migrated from a legacy product. Migration is performed by an external service. All you have to do for a request that needs migration is to skip processing by raising a `connect.exceptions.SkipRequest` exception.

Parameters **migration_key** (*str*) – The name of the parameter that contains the migration data (optional; default value is `migration_info`).

Returns Whether the request needs migrating.

Return type bool

new_items

Returns New items.

Return type List[*Item*]

note = None
(str) Details of note.

params_form_url = None
(str) URL for customer/reseller/provider for modifying param value based on vendor's feedback.

reason = None
(str) Fail reason in case of status of request is failed.

removed_items

Returns Removed items.

Return type List[*Item*]

status = None
(str) Status of request. One of:

- pending
- inquiring
- failed
- approved

Valid status changes:

- pending -> inquiring
- pending -> failed
- pending -> approved
- inquiring -> failed
- inquiring -> approved
- inquiring -> pending

type = None
(str) Asset status. See `Asset` class for details.

updated = None
(datetime.datetime) Date of last request modification.

class connect.models.**Hub**(***kwargs*)
A Hub.

company = None
(`Company`) Reference to the company the hub belongs to.

description = None
(str|None) Hub description (Markdown text).

events = None
(`Events`) Events occurred on Hub.

instance = None
(`HubInstance`) Hub instance.

name = None
(str) Hub name.

stats = None
(`HubStats`) Hub stats.

class connect.models.**HubInstance**(***kwargs*)
An instance of a hub.

type = None
(str) E-Commerce system type.

class connect.models.**HubStats**(***kwargs*)
Hub stats.

connections = None
(int) Number of connections active for this Hub.

marketplaces = None
(int) Number of marketplaces for this Hub.

class connect.models.**Item**(***kwargs*)
A product item.

global_id = None
(str) Global id.

mpn = None
(str) Item manufacture part number.

old_quantity = None
(int|float|None) Previous value of quantity.

quantity = None
(int|float) Number of items of the type in the asset (-1 if unlimited)

renewal = None
(*Renewal* | None) Parameters of renewal request (empty for all other types).

class connect.models.**Marketplace**(***kwargs*)
An object containing Distribution agreements with exact Hubs, enriched with additional information on details about the relation.

A Marketplace is a way to list Products to specified regions (based on Distribution Agreements) and use specific Hubs to provision incoming Fulfillment requests.

active_contracts = None
(int) How many active contracts were signed on the Marketplace.

description = None
(str) Markdown text describing the marketplace.

hubs = None
(List[*ExtIdHub*]) List of account-hub relations associated with the Marketplace object.

icon = None
(str) Image identifying Marketplace object uploaded by user.

name = None
(str) Marketplace title, unique for an account.

owner = None
(*Company*) Provider account - the object owner.

zone = None
(str) Zone where the marketplace is located, there can be following zones: AF, NA, OC, AS, EU, SA (It is continents).

class connect.models.**Param**(***kwargs*)
Parameters are used in product and asset definitions.

constraints = None
(*Constraints* | None) Parameter constraints.

description = None
(str) Description of parameter.

name = None
(str) Name of parameter.

scope = None
(str|None) Scope of parameter.

title = None
(str|None) Title for parameter.

type = None
(str) Type of parameter.

value = None
(str|None) Value of parameter.

value_choice = None
(List[str]|None) Available choices for parameter.

value_error = None
(str|None) Error indicated for parameter.

class connect.models.**PhoneNumber**(***kwargs*)
Phone number.

area_code = None
(str|None) Area code.

country_code = None
(str|None) Country code.

extension = None
(str|None) Phone extension.

phone_number = None
(str|None) Phone number.

class connect.models.**Product**(***kwargs*)
Represents basic marketing information about salable items, parameters, configurations, latest published version and connections.

It contains basic product information like name, description and logo, along with the latest published version details. So in a single point we can say a single product object always represent the latest published version of that product.

configurations = None
(*ProductConfiguration*) Product configuration.

customer_ui_settings = None
(*CustomerUiSettings*) Customer Ui Settings.

detailed_description = None
(str) Detailed description of product.

icon = None
(str) Product icon URI.

name = None
(str) Product name.

short_description = None
(str) Short description of product.

version = None
(int) Version of product.

class connect.models.**ProductConfiguration**(***kwargs*)
Product configurations.

requires_reseller_information = None
(bool) Does the product require reseller information?

suspend_resume_supported = None
(bool) Is suspend and resume supported for the product?

class connect.models.**Renewal**(***kwargs*)
Item renewal data.

from_ = None
(datetime.datetime) Date of renewal beginning.

period_delta = None
(int) Size of renewal period.

period_uom = None
(str) Unit of measure for renewal period. One of: year, month, day, hour.

to = None
(datetime.datetime) Date of renewal end.

class connect.models.**ServerErrorResponse**(***kwargs*)
Server response when an error occurs.

error_code = None
(str) Error code.

errors = None
(List[str]) List of errors.

params = None
(dict) Error params.

class connect.models.**Template**(***kwargs*)
Tier Template

representation = None
(str) Template representation.

class connect.models.**TierAccount**(***kwargs*)
Tier account.

contact_info = None
(*ContactInfo*) Tier Contact Object.

external_id = None
(str|None) Only in case of filtering by this field.

external_uid = None
(str|None) Only in case of filtering by this field.

name = None
(str) Tier name.

class connect.models.**TierAccounts**(***kwargs*)
TierAccounts object.

customer = None
(*TierAccount*) Customer Level TierAccount Object.

tier1 = None
(*TierAccount*) Level 1 TierAccount Object.

tier2 = None
(*TierAccount*) Level 2 TierAccount Object.

class connect.models.**TierConfig**(***kwargs*)
Full representation of Tier object.

account = None
(*TierAccount*) Full tier account representation (same as in *Asset*).

connection = None
(*Connection*) Reference to Connection Object.

events = None
(*Events* | None) Tier Config events.

classmethod get(*tier_id*, *product_id*, *config=None*)

Gets the specified tier config data. For example, to get Tier 1 configuration data for one request we can do:

```
TierConfig.get(request.asset.tiers.tier1.id, request.asset.product.id)
```

Parameters

- **tier_id** (*str*) – Id of the requested Tier Config.
- **product_id** (*str*) – Id of the product.
- **config** (Config) – Config to use, or None to use environment config (default).

Returns The requested Tier Config, or None if it was not found.

Return type Optional[*TierConfig*]

get_param_by_id(*id_*)

Get a Tier Config parameter.

Parameters **id** (*str*) – Parameter id.

Returns The requested parameter, or None if it was not found.

Return type *Param*

name = None

(str) Tier configuration of account.name.

open_request = None

(*BaseModel* | None) Reference to TCR.

params = None

(List[*Param*]) List of TC parameter data objects as in Asset Object extended with unfilled parameters from product.

product = None

(*Product*) Reference object to product (application).

template = None

(*Template*) Template Object.

tier_level = None

(int) Tier level for product from customer perspective.

class connect.models.**TierConfigRequest**(***kwargs*)

activation = None

(*Activation* | None) Activation object. This is created only if TCR has ordering parameters and seen in inquiring state of the TCR.

assignee = None

(*User* | None) TCR environment. One of: test, prod, preview.

configuration = None

(*TierConfig*) Full representation of Tier Configuration Object.

events = None

(*Events* | None) Tier Config request Events.

get_param_by_id(*id_*)

Get a Tier Config Request parameter.

Parameters **id** (*str*) – Parameter id.

Returns The requested parameter, or `None` if it was not found.

Return type *Param*

notes = None
(str) TCR pending notes. Notes can be modified only in Pending state.

params = None
(List[*Param*]) List of parameter data objects as in Asset Object. Params can be modified only in Pending state.

reason = None
(str|None) Failing reason. This is filled only if TCR is failed.

status = None
(str) TCR current status. One of: tiers_setup, pending, inquiring, approved, failed.

template = None
(*Template* | None) Template Object. This is filled only if TCR is approved.

type = None
(str) TCR type. One of: setup, update.

class connect.models.**UsageFile**(***kwargs*)
Usage File Object.

acceptance_note = None
(str) Note provided by the provider in case of acceptance of the usage file.

accepted_at = None
(str) Usage file acceptance time.

accepted_by = None
(str) User Id, who accepted the usage file as provider.

closed_at = None
(str) Usage file billing time.

closed_by = None
(str) User Id, who billed the usage file as provider.

contract = None
(*Contract*) Reference on Contract Object.

created_at = None
(str) Date of the creation of the UsageFile.

created_by = None
(str) User ID who have created this UsageFile.

description = None
(str) Vendor can provide a description value in this field to describe the file content.

error_details = None
(str) In case of invalid file, this field will contain errors related to the file.

marketplace = None
(*Marketplace* Reference on Marketplace Object.

name = None
(str) Name of the Usage file object.

note = None
(str) Vendor can put a note which can be refer later for some extra information.

processed_file_uri = None
(str) Google Storage shared location of the generated file after processing uploaded file. Only available in GET API and not included in list API (sharing timeout 30 sec).

product = None
(*Product*) Reference on Product Object.

provider = None
(*Company* Reference object to the provider company.

records = None
(*UsageRecords*) UsageRecords Object.

rejected_at = None
(str) Usage file rejection time.

rejected_by = None
(str) User Id, who rejected the usage file as provider.

rejection_note = None
(str) Note provider by the provider in case of rejection of the usage file.

status = None
(str) One of: draft, uploading, uploaded, processing, invalid, ready, rejected, pending, accepted, closed.

submitted_at = None
(str) User Id, who submitted the usage record to Provider.

submitted_by = None
(str) User Id, who submitted the usage record to Provider.

upload_file_uri = None
(str) Google Storage shared location of the upload file. Only available in GET API and not included in list API (sharing timeout 600 sec).

uploaded_at = None
(str) Usage file upload time.

uploaded_by = None
(str) User Id who uploaded the file.

vendor = None
(*Company* Reference object to the vendor company.

class connect.models.**UsageListing**(***kwargs*)
Usage Listing Object.

contract = None
(*Contract*) Contract Object.

created = None
(str) Creation time.

product = None
(*Product*) Product Object.

provider = None
(*Company*) Provider Object.

status = None
(str) Status.

vendor = None
(*Company*) Vendor Object.

class connect.models.**UsageRecord**(***kwargs*)
Usage Record Object.

asset_search_criteria = None
(str) Asset search criteria.

asset_search_value = None
(str) Asset search value.

end_time_utc = None
(str) End Time in UTC.

item_search_criteria = None
(str) Item search criteria.

item_search_value = None
(str) Item search value.

quantity = None
(int) Quantity.

record_id = None
(str) Record id.

start_time_utc = None
(str) Start Time in UTC.

class connect.models.**UsageRecords**(***kwargs*)
Usage Records Object.

invalid = None
(int) Invalid.

valid = None
(int) Valid.

class connect.models.**User**(***kwargs*)
Represents a user within the platform.

name = None
(str) User name.

class connect.models.**ValueChoice**(***kwargs*)
A value choice for a parameter.

label = None
(str) Label.

value = None
(str) Value.

2.2.4 resources

class connect.resources.**FulfillmentAutomation**(*config=None*)
This is the automation engine for the Fulfillment API. If you want to process fulfillment requests, subclass this and implement the `process_request` method, which receives a *connect.models.Fulfillment* request as argument and must return an *connect.models.ActivationTemplateResponse* or *connect.models.ActivationTileResponse* object in case the request has to be approved.

In other case, you must raise one of these exceptions:

- `connect.exceptions.InquireRequest`: Inquire for more information.
- `connect.exceptions.FailRequest`: Causes the request to fail.
- `connect.exceptions.SkipRequest`: Skips processing the request.

Create an instance of your subclass and call its `process` method to begin processing.

For an example on how to use this class, see *Processing Fulfillment Requests*.

`filters`(*status='pending'*, ***kwargs*)

Parameters

- **status** (*str*) – Status of the requests. Default: `'pending'`.
- **kwargs** (*dict[str,Any]*) – Additional filters to add to the default ones.

Returns The set of filters for this resource.

Return type dict[str,Any]

`get_tier_config`(*tier_id*, *product_id*)

Gets the specified tier config data. For example, to get Tier 1 configuration data for one request, within the FulfillmentAutomation instance, we can do:

```
self.get_tier_config(request.asset.tiers.tier1.id, request.asset.product.id)
```

Parameters

- **tier_id** (*str*) – Id of the requested Tier Config.
- **product_id** (*str*) – Id of the product.

Returns The requested Tier Config, or `None` if it was not found.

Return type Optional[*TierConfig*]

Deprecated since version 16.0: Use `TierConfig.get` instead.

`model_class`

alias of `connect.models.fulfillment.Fulfillment`

`update_parameters`(*pk*, *params*)

Sends a list of Param objects to Connect for updating.

Parameters

- **pk** (*str*) – Id of the request.
- **params** (*list[Param]*) – List of parameters to update.

Returns The server response.

Return type str

class `connect.resources.`**`TemplateResource`**(*config=None*)

Template Resource.

`get`(*pk*)

Get an activation template.

Parameters **pk** (*str*) – Primary key of the template to obtain.

Returns ActivationTemplateResponse object with template contents.

Return type *ActivationTemplateResponse*

render(*pk*, *request_id*)

Get an activation tile.

Parameters

- **pk** (*str*) – Primary key of the template to obtain.
- **request_id** (*str*) – Id of the associated request.

Returns ActivationTileResponse object with tile contents.

Return type *ActivationTileResponse*

class connect.resources.**TierConfigAutomation**(*config=None*)

This is the automation engine for the Tier Config Request API. If you want to process Tier Config requests, subclass this and implement the `process_request` method, which receives a *connect.models.TierConfigRequest* request as argument and must return an *connect.models.ActivationTemplateResponse* or *connect.models.ActivationTileResponse* object in case the request has to be approved.

In other case, you must raise one of these exceptions:

- *connect.exceptions.InquireRequest*: Inquire for more information.
- *connect.exceptions.FailRequest*: Causes the request to fail.
- *connect.exceptions.SkipRequest*: Skips processing the request.

Create an instance of your subclass and call its `process` method to begin processing.

For an example on how to use this class, see *Processing Tier Config Requests*.

model_class

alias of `connect.models.tier_config.TierConfigRequest`

update_parameters(*pk*, *params*)

Sends a list of Param objects to Connect for updating.

Parameters

- **pk** (*str*) – Id of the request.
- **params** (*list[Param]*) – List of parameters to update.

Returns The server response.

Return type str

class connect.resources.**UsageAutomation**(*config=None*)

Automates reporting of Usage Files.

For an example on how to use this class, see *Reporting Usage Files*.

filters(*status='listed'*, ***kwargs*)

Parameters

- **status** (*str*) – Status of the requests. Default: `'listed'`.
- **kwargs** (*dict[str,Any]*) – Additional filters to add to the default ones.

Returns The set of filters for this resource.

Return type dict[str,Any]

get_usage_template(*product*)

Returns the template file contents for a specified product.

Parameters **product** (Product) – Specific product.

Returns The template file contents.

Return type bytes

Raises ***FileRetrievalError*** – Raised if the file contents could not be retrieved.

model_class

alias of `connect.models.usage.UsageFile`

submit_usage(*usage_file*, *usage_records*)

Submit a usage file.

Parameters

- **usage_file** (UsageFile) – Usage file.
- **usage_records** (*list* [UsageRecord]) – Records.

Returns Usage file.

Return type *UsageFile*

Raises ***FileCreationError*** – Raised if creation or uploading of the file fails.

class connect.resources.**UsageFileAutomation**(*config=None*)

Automates workflow of Usage Files.

For an example on how to use this class, see *Workflow of Usage Files*.

model_class

alias of `connect.models.usage.UsageFile`

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index