Import Transactions

Example of how automate daily import of Transactions for Client Portfolios

`com/finmars/client-daily-schedule/import-transactions/workflow.json`

```
{
    "workflow":
    {
        "user_code":"com.finmars.client-daily-schedule:import-transactions",
        "is_manager": true,
        "tasks": ["com.finmars.client-daily-schedule:import-transactions.task"],
        "periodic": {
            "crontab": "0 2 * * *"
        }
    }
}
```

`com/finmars/client-daily-schedule/import-transactions/tasks.py`

This simple code basically does

- get transactions
- transform transactions
- import transactions

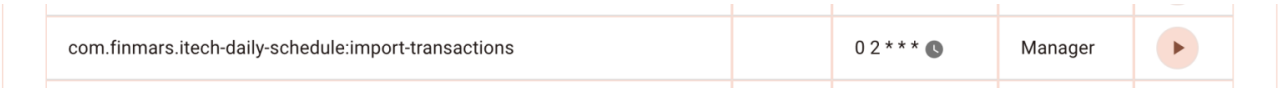
```
import jwt
import copy
import json
import requests
from datetime import datetime, timezone, timedelta
from requests.auth import HTTPBasicAuth
import traceback
import time

from workflow.api import task
from workflow.finmars import storage, utils, vault, request_api, create_logger
```

```
workflow_user_code="com.finmars.client-daily-schedule:import-transactions"

@task(name="%s.task" %workflow_user_code, bind=True)
def main(self, *args, **kwargs):



    # Expected payload
    # payload = {
    #    "date": "2024-09-23"
    # }



    # TODO we need connection between Portfolio and Client and Vault Keypath, right now
hardcoded
    # portfolios = get_portfolios()

    # Get today's date
    today = datetime.now()

    # Calculate yesterday's date
    yesterday = today - timedelta(days=1)

    payload = kwargs.get("payload", {})

    target_date = payload.get('date', str(yesterday.date()))

    portfolios = [
        {
        "secret": "finmars/exante",
        "user_code": "ABCD123.001"
        },
    ]

    self.log(f"Target Date {target_date}")

    for item in portfolios:

        workflow_get_positions = self.workflow.run_new_workflow(
```

```
            "com.finmars.exante-broker:get-transactions",
            payload={
            "secret": item['secret'],
            "date_from": target_date,
            "date_to": target_date,
            "portfolios": [item['user_code']]
            }
        )

        poll_workflow_status(workflow_get_positions['id'])

        self.log(f"Transactions Received from Exante for {item['user_code']}")

        workflow_transform_positions = self.workflow.run_new_workflow(
            "com.finmars.exante-data-transformer:transform-transactions",
            payload={
            "date_from": target_date,
            "date_to": target_date,
            "portfolios": [item['user_code']]
            }
        )

        poll_workflow_status(workflow_transform_positions['id'])

        self.log(f"Transactions Transformed to Standard Configuration for
{item['user_code']}")

        workflow_import_positions = self.workflow.run_new_workflow(
            "com.finmars.standard-workflow:simple-autoimport",
            payload={
                "actions": [
                    {
                        "file_path":
f"/data/general/transactions/exante_{convert_to_ascii(item['user_code'])}_{target_date}_{targe
t_date}.json",
                        "import_type": "transaction",
                        "schema": "com.finmars.standard-import-from-file:txn"
                    }
                ]
            }
```

```
        )

        poll_workflow_status(workflow_transform_positions['id'])

        self.log(f"Transactions Imported for {item['user_code']}")

# should be moved to Finmars Workflow Library
def poll_workflow_status(workflow_id, max_retries=100, wait_time=5):
    url = f'/workflow/api/workflow/{workflow_id}/'  # Replace with your actual API endpoint

    for attempt in range(max_retries):
        data = request_api(url)

        if data:
            status = data.get('status')
            print(f'Attempt {attempt + 1}: Workflow status is {status}')

            if status in ['success', 'error']:
                return status  # Return the status when it's success or error

        else:
            print(f'Error fetching status: {response.status_code}')

        time.sleep(wait_time)  # Wait before the next attempt

    print('Max retries reached. Workflow status not successful.')
    return None  # Indicate that the status was not found


def convert_to_ascii(input_string):

    input_string = input_string.lower()

    # Convert spaces and dots to underscores
    modified_string = input_string.replace(' ', '_').replace('.', '_')

    # Convert the string to ASCII, ignoring non-ASCII characters
    ascii_string = modified_string.encode('ascii', 'ignore').decode()

    return ascii_string
```

In the end you should able to see your workflow Scheduled in Finmars Workflow



Revision #10
Created 2024-09-30 23:22:25 UTC by Sergei Zhitenev
Updated 2024-09-30 23:46:15 UTC by Sergei Zhitenev