Daily Import

- Daily Import Introduction
- Import Positions
- Import Transactions
- Import Prices
- Import FX Rates

Daily Import Introduction

What is Daily Import?

Its a pipeline that covers all workflows and schedules that runs in background to import all the data into your Space in Finmars

Daily Import could be different from client to client, from provider to provider, but as core concept it should have these essential steps:

- Daily Positions Import
  - Download Positions
  - Transform Postions to Standard Format
  - Parse Positions to find new Instruments/Currencies
  - Import new Instruments/Currencies
  - Import Positions
- Daily Transactions Import
  - Download Transactions
  - Transform Transactions to Standard Format
  - Import Transactions
- Valuations Import
  - Download Prices
  - Transform Prices to Standard Format
  - Import Prices
  - Download FX Rates
  - Transform FX Rates to Standard Format
  - Import FX Rates

Thats a Basic ETL flow that should be applied on Daily Basis

Import Positions

Example of how automate daily import for Client Portfolios

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

```
{
    "workflow":
    {
        "user_code":"com.finmars.client-daily-schedule:import-positions",
        "is_manager": true,
        "tasks": ["com.finmars.client-daily-schedule:import-positions.task"],
        "periodic": {
            "crontab": "0 1 * * *"
        }
    }
}
```

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

This simple code basically does

- get positions
- transform positions
- import positions

```
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-positions"

@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-positions",
        payload={
        "secret": item['secret'],
        "date_from": target_date,
        "date_to": target_date,
        "currencies": ["USD"],
        "portfolios": [item['user_code']]
        }
    )

    poll_workflow_status(workflow_get_positions['id'])

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


    workflow_get_positions = self.workflow.run_new_workflow(
        "com.finmars.exante-broker:get-instruments",
        payload={
        "secret": item['secret'],
        "date_from": target_date,
        "date_to": target_date,
        "currencies": ["USD"],
        "portfolios": [item['user_code']]
        }
    )

    poll_workflow_status(workflow_get_positions['id'])

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

    workflow_get_positions = self.workflow.run_new_workflow(
        "com.finmars.exante-data-transformer:transform-instruments",
        payload={}
    )

    poll_workflow_status(workflow_get_positions['id'])

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

    directories, files = storage.listdir('/data/general/instruments/')
```

```
        if len(files):
            for file in files:
                workflow_get_positions = self.workflow.run_new_workflow(
                    "com.finmars.standard-workflow:simple-autoimport",
                    payload={
                        "actions": [
                            {
                                "file_path": f"/data/general/instruments/{file}",
                                "import_type": "data",
                                "schema": "com.finmars.standard-import-from-
file:instruments.instrument:instruments"
                            }
                        ]
                    }
                )

                poll_workflow_status(workflow_get_positions['id'])

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


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

        poll_workflow_status(workflow_transform_positions['id'])

        self.log(f"Position 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/positions/exante_{convert_to_ascii(item['user_code'])}_{target_date}_{target_d
ate}.json",
                        "import_type": "transaction",
                        "schema": "com.finmars.standard-import-from-file:pos_daily"
                    }
                ]
            }
        )

        poll_workflow_status(workflow_transform_positions['id'])

        self.log(f"Position 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

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

com.finmars.itech-daily-schedule:import-transactions		0 2 * * *	Manager	

Import Prices

Example of how automate daily import of Prices for Client Portfolios

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

```
{
    "workflow":
    {
        "user_code":"com.finmars.client-daily-schedule:import-prices",
        "is_manager": true,
        "tasks": ["com.finmars.client-daily-schedule:import-prices.task"],
        "periodic": {
            "crontab": "30 2 * * *"
        }
    }
}
```

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

This simple code basically does

- get prices
- transform prices
- import prices

```
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-prices"

@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()))

    instruments = get_instruments()

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

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

```
    workflow_get_positions = self.workflow.run_new_workflow(
        "com.finmars.exante-broker:get-prices",
        payload={
        "secret": portfolios[0]['secret'],
        "date_from": target_date,
        "date_to": target_date,
        "instruments": instruments
        }
    )

    poll_workflow_status(workflow_get_positions['id'])

    self.log(f"Prices Received from Exante")

    workflow_transform_positions = self.workflow.run_new_workflow(
        "com.finmars.exante-data-transformer:transform-prices",
        payload={
        "date_from": target_date,
        "date_to": target_date,
        "instruments": instruments
        }
    )

    poll_workflow_status(workflow_transform_positions['id'])

    self.log(f"Prices Transformed to Standard Configuration")

    for instrument in instruments:
        workflow_import_positions = self.workflow.run_new_workflow(
            "com.finmars.standard-workflow:simple-autoimport",
            payload={
                "actions": [
                    {
                        "file_path":
f"/data/general/prices/exante_{convert_to_ascii(instrument)}_{target_date}_{target_date}.json"
,
                        "import_type": "data",
                        "schema": "com.finmars.standard-import-from-
file:instruments.pricehistory:price"
```

```
                    }
                ]
            }
        )

        # poll_workflow_status(workflow_transform_positions['id'])

        self.log(f"Prices Imported")

# 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 get_instruments():

    data = request_api('/api/v1/instruments/instrument/light/')

    instruments = []

    for item in data['results']:
        if item['short_name'] != '-':
```

```
            instruments.append(item['short_name'])

    return instruments

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 able to see you workflow Scheduled in Finmars Workflow

com.finmars.itech-daily-schedule:import-prices		30 2 * * *	Manager	

Import FX Rates

Just make sure you have installed `com.finmars.finmars-database-fx-rates` **Finmars Database FX Rates**

You can change Schedule in workflow.json crontab

`com/finmars/finmars-database-fx-rates/import-fx-rates/workflow.json`

```
{
    "workflow":
    {
        "user_code":"com.finmars.finmars-database-fx-rates:import-fx-rates",
        "is_manager": false,
        "tasks": ["com.finmars.finmars-database-fx-rates:import-fx-rates.task"],
        "periodic": {
            "crontab": "0 3 * * *"
        }
    }
}
```

So in the end you need to see your definition scheduled in Finmars Workflow