Keywords

Workflow - some process defined by user, it sequence of one step or many steps that do something, usually its run by some **Schedule**. Job, Pipeline are all related terms

Workflow Engine - a tool, Django app created by Finmars SCSA that helps to create and execute **Workflow**. This service is absolutely separated from Finmars Platform (backend) due to security and logical concerns.

Schedule - part of **Workflow Engine**, this service allows to execute **Workflow** on some regular schedule (with crontab) and user can define **Payload** to that Schedule as well

Payload - an input of your **Workflow**, normally its a JSON object (dictionary) that will be passed into Workflow Code. Its accessible via *kwargs.get("payload", None)* Payload itself is user-defined as its workflows

Definition - old keyword, but basically its a YAML/JSON document that describes Workflow itself (meta-file)

Workflow (version 1) - First Generation of Worfklows, its defined in *worfklow.json.* This types are still supported, but deprecated. Consider using Workflow (version 2)

Example of its structure (JSON)

```
{
   "workflow":
   {
      "user_code":"com.finmars.standard-workflow:simple-autoimport",
      "is_manager": false,
      "tasks": ["com.finmars.standard-workflow:simple-autoimport.task"]
   }
}
```

Example of its structure (YAML)

```
workflow:
   user_code: com.finmars.standard-workflow:collect-price
   is_manager: false
   tasks:
     - com.finmars.standard-workflow:collect-price.task
```

Workflow (version 2) - Second Generation of Workflows, its defined in workflow.json. This structure is more complex and generated by Workflow Engine itself. But user still able to tune it manually. (see Workflow Template)

Workflow Engine look at "*version*": property, so "*version*": "*2*", is for **Workflow** (version 2). If not defined it will be **Workflow** (version 1)


```
{
   "version": "2",
   "workflow": {
      "name": "Nested Test",
      "user_code": "com.finmars.local:nested",
      "default_payload": {
         "report_date": "2024-10-10"
      },
      "nodes": [
         {
            "id": "b33a5bc711c739b7",
            "data": {
               "node": {
                  "name": "step-1",
                  "type": "workflow",
                  "notes": "",
                  "user_code": "step-1"
               },
               "workflow": {
                  "name": "Download Data",
                  "user_code": "com.finmars.example-etl:download"
               },
               "source_code": "\ndef main(self, *args, **kwargs):\n    self.log(\"Hello World\")\n\n    return {\"status\": \"success\"}\n"
            },
            "name": "step-1",
            "label": "",
            "width": 422,
            "height": 598,
            "inputs": {
               "in": {
                  "id": "3f1c9322cca914ed",
                  "label": "Input",
```

```
            "socket": {
               "name": "socket"
            },
            "control": null,
            "showControl": true
         },
         "payload_input": {
            "id": "caf1ac1948c12178",
            "label": "Payload",
            "socket": {
               "name": "payload_socket"
            },
            "control": null,
            "showControl": true
         }
      },
      "outputs": {
         "out": {
            "id": "57eee5fd73fec7e5",
            "label": "Output",
            "socket": {
               "name": "socket"
            },
            "multipleConnections": true
         }
      },
      "controls": {},
      "position": {
         "x": -361.12865172833614,
         "y": -374.3578764111789
      }
   },
   {
      "id": "b5744926f4976461",
      "data": {
         "node": {
            "name": "conditional-test",
            "type": "workflow",
            "notes": "",
            "user_code": "conditional-test"
```

```
            },
            "workflow": {
               "name": "Test Vault",
               "user_code": "com.finmars.local:test-vault"
            },
            "source_code": "\ndef main(self, *args, **kwargs):\n    self.log(\"Hello World\")\n\n    return {\"status\": \"success\"}\n"
         },
         "name": "conditional-test",
         "label": "",
         "width": 417,
         "height": 719,
         "inputs": {
            "in": {
               "id": "5cee94f3086e8e3e",
               "label": "Input",
               "socket": {
                  "name": "socket"
               },
               "control": null,
               "showControl": true
            },
            "payload_input": {
               "id": "1e9f1cdb08f3b2e2",
               "label": "Payload",
               "socket": {
                  "name": "payload_socket"
               },
               "control": null,
               "showControl": true
            }
         },
         "outputs": {
            "out": {
               "id": "7ea2cdd214c5ba7e",
               "label": "Output",
               "socket": {
                  "name": "socket"
               },
               "multipleConnections": true
```

```
            }
          },
          "controls": {},
          "position": {
            "x": 423.5227311216107,
            "y": -488.5574374671881
          }
        },
        {
          "id": "4324a1babd889f5f",
          "data": {
            "node": {
              "name": "final-step",
              "type": "source_code",
              "notes": "",
              "user_code": "final-step"
            },
            "workflow": {
              "name": "Test Vault",
              "user_code": "com.finmars.local:test-vault"
            },
            "source_code": "\ndef main(self, *args, **kwargs):\n    self.log(\"Hello World\")\n\n    return {\"status\": \"success\"}\n"
          },
          "name": "final-step",
          "label": "",
          "width": 440,
          "height": 790,
          "inputs": {
            "in": {
              "id": "969915a7f2d9fcf2",
              "label": "Input",
              "socket": {
                "name": "socket"
              },
              "control": null,
              "showControl": true
            },
            "payload_input": {
              "id": "2858c38d5fc4a37b",
```

```
                "label": "Payload",
                "socket": {
                    "name": "payload_socket"
                },
                "control": null,
                "showControl": true
            }
        },
        "outputs": {
            "out": {
                "id": "694544a7e2761259",
                "label": "Output",
                "socket": {
                    "name": "socket"
                },
                "multipleConnections": true
            }
        },
        "controls": {},
        "position": {
            "x": 1050.2581884865385,
            "y": -536.2804491695036
        }
    },
    {
        "id": "5de8268354a95a58",
        "data": {
            "node": {
                "name": "step-2-payload-generator",
                "type": "source_code",
                "notes": "",
                "user_code": "step-2-payload-generator"
            },
            "workflow": {},
            "source_code": "\ndef main(self, *args, **kwargs):\n    self.log(\"Hello World\")\n    \n    payload = kwargs.get(\"payload\")\n    \n    # options = payload.get(\"options\")\n    \n    # if self.user_code in options['options_steps']:\n    #     return\n    \n    \n    # execution continue\n    return {\"hello\": \"world\"}\n"
        },
        "name": "step-2-payload-generator",
        "label": "",
```

```
        "width": 316,
        "height": 696,
        "inputs": {
          "in": {
            "id": "1d0df286dd91ce9c",
            "label": "Input",
            "socket": {
              "name": "socket"
            },
            "control": null,
            "showControl": true
          },
          "payload_input": {
            "id": "f56cfc9b414c599d",
            "label": "Payload",
            "socket": {
              "name": "payload_socket"
            },
            "control": null,
            "showControl": true
          }
        },
        "outputs": {
          "out": {
            "id": "4c9d99bde3f8f940",
            "label": "Output",
            "socket": {
              "name": "socket"
            },
            "multipleConnections": true
          }
        },
        "controls": {},
        "position": {
          "x": 1.1510722982846584,
          "y": -1111.1812663772207
        }
      }
    ],
    "connections": [
```

```
        {
          "sourceOutput": "out",
          "targetInput": "in",
          "id": "36b8b8ebc0bcd3d9",
          "source": "b33a5bc711c739b7",
          "target": "b5744926f4976461"
        },
        {
          "sourceOutput": "out",
          "targetInput": "in",
          "id": "9560ed0b2b5092d3",
          "source": "b5744926f4976461",
          "target": "4324a1babd889f5f"
        },
        {
          "sourceOutput": "out",
          "targetInput": "payload_input",
          "id": "763afddc95919939",
          "source": "5de8268354a95a58",
          "target": "b5744926f4976461"
        }
      ],
      "comments": []
    }
}
```

To simplify structure it will be like:

```
{
  "version": "2",
  "workflow": {
    "name": "Nested Test",
    "user_code": "com.finmars.local:nested",
    "default_payload": {
      ... # predefined payload
    },
    "nodes": [
      ...  # steps of workflow
```

```
        ],
        "connections": [
          ... # connection between steps (defines order of execution)
        ],
        "comments": []
    }
}
```

Workflow Step - it can be either **Task** or **Workflow** (yes, workflow could be nested)

Task - actual code of Workflow that will be executed, right now Workflow Engine supports only Python,

Status - indicator of what is going with Task/Workflow, possible statuses:

- init - task is created, but awaits its execution
- progress - task is executing
- nested-progress - same as progress, but for nested scenarios
- success - task finishes successfully
- error - something went wrong
- timeout - Workflow Engine stops a task due long execution (normally its 86400 seconds or 24hours)
- canceled - User stops a task for some reason
- wait - special status for Workflow, workflow can be set on **Pause**, so all tasks in progress will be finished and no new tasks is being executed until user **Resume** Workflow

Parent Workflow - so, each **Task** is part (child) of **Workflow**, that means Workflow receive status Success when all of its child Tasks will be success. Also, because Nested Workflow is also wrapped as a Task, it receives status of nested-progress instead of progress.

Explorer - part of Finmars Platform Services, it is a File Storage, so its used not only to store user files (like documents or invoices) but store Data from Data Providers and also is a place where Source Code of Workflows is stored. Normally it stored in path */workflows/%module_name/%workflow_name%/* (e.g. *workflows/com/finmars/standard-workflow/collect-prices*)

Configuration Code - Unique identifier of **Configuration Module**, this modules shared across Finmars **Spaces** via Finmars **Marketplace.** Its a reverse DNS pattern, e.g. com.finmars.standard-workflow.

Configuration Module - a package with some extension to Finmars Platform, normally it some Meta Files that extend configuration Platform (e.g. Transaction Types, Import Schemes) But Configuration Module also could contains Source Code of Worfklows.

Marketplace - users able to create own Workflows that solve certain problems (e.g. integration to some Broker/Bank, or extending Finmars Pricing Engine, or even create some custom apps (e.g. PDF Builder)) and share it with other people via Marketplace. Source Code packed as Configuration Module and it can be installed later with single click

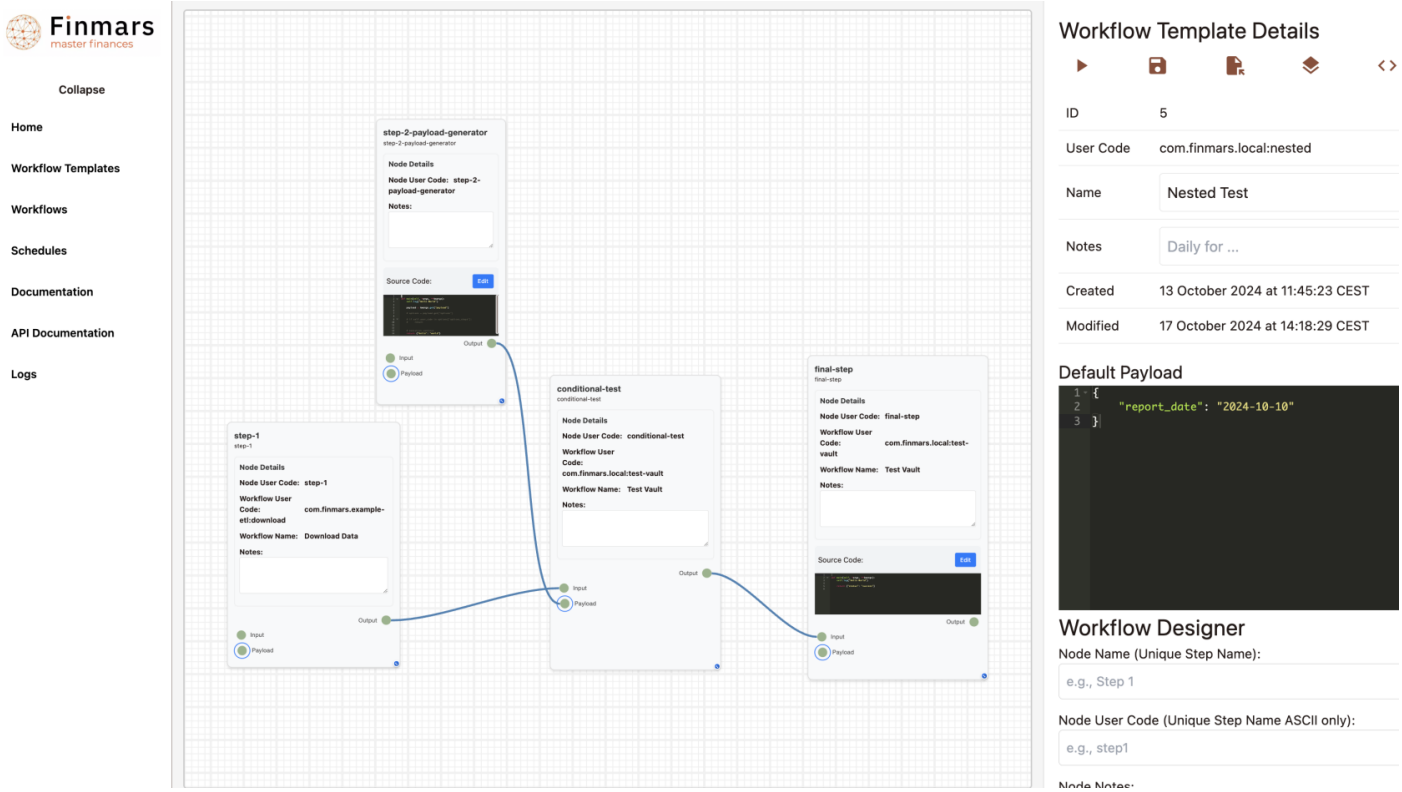
Task (Version 1) - example of how to register your task in Workflow (Version 1)

```
import json
import requests

from workflow.api import task

workflow_user_code="com.finmars.standard-workflow:hello-world"

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

  print("Hello World")
```

So you must define main function and wrap it with decorator from *from workflow.api import task*.

Also you feel free to import and modules in your python code. See list of available modules here.

Workflow Template - is a Workflow (Version 2), so instead of just using YAML/JSON structure Workflow Engine provides user with Visual Editor of Workflows



So user able to create new Steps and drop them on the pane and connect Steps with each other (defining order of execution)
Available Steps Types are: Workflow (Version 1), Custom Code, Workflow (Version 2)

Revision #9
Created 3 November 2024 11:14:29 by Sergei Zhitenev
Updated 3 November 2024 12:01:47 by Sergei Zhitenev