

Flexible DAG Trigger Forms (AIP-50)

Jens Scheffler
Christian Schilling



Triggering a DAG via UI – Is this a Problem?



Product Owner

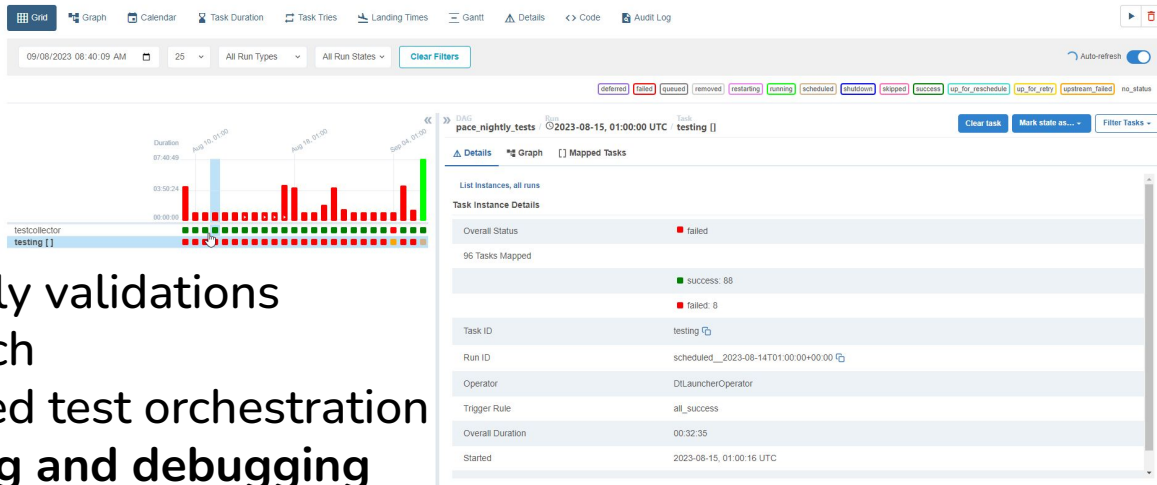


Technical lead

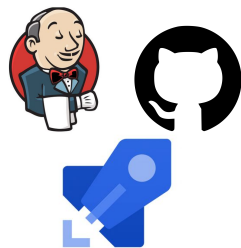
Open loop testing for verification and validation of highly automated driving functions

Use cases:

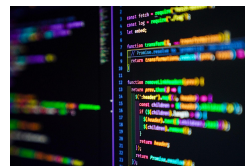
- Automated and scaled nightly validations
- Smoke testing on main branch
- Hardware and software based test orchestration
- Parametrized manual testing and debugging



Triggering a DAG via UI – The Past



Apache
Airflow



Airflow DAGs Security Browse Admin Docs Test Plugin

DAGs

All 32 Active 1 Paused 31 Filter DAGs by tag

DAG	Owner	Runs	Schedule	Last Run
<input type="checkbox"/> example_bash_operator		airflow	0 0 ***	

Triggering a DAG via UI – The Basics



Airflow

DAGs

Security

Browse

Admin

Docs

Launch Job

19:15 UTC

AF

DAG Trigger UI - Example Batch Job Run

Job Parameters

GIT Branch *:

main

GIT branch name (or GIT commit hash) which the job should be executed

File List *:

/path/to/one/file1.data
/path/to/one/file2.data
/path/to/one/file3.data

List of files (path references) which should be executed the job on, one line per file

System type to test *:

System B

System A

System B

Triggering a DAG via UI – The Basics

```
with DAG(  
    dag_id=Path(__file__).stem,  
    description=__doc__.partition(".")[0],  
    doc_md=__doc__,  
    schedule=None,  
    start_date=datetime.datetime(2022, 3, 4),  
    catchup=False,  
    tags=["example_ui"],  
    params={  
        "names": Param(  
            ["Linda", "Martha", "Thomas"],  
            type="array",  
            description="Define the list of names for which greetings should be generated in the logs."  
            " Please have one name per line.",  
            title="Names to greet",  
        ),  
        "english": Param(True, type="boolean", title="English"),  
        "german": Param(True, type="boolean", title="German (Formal)"),  
        "french": Param(True, type="boolean", title="French"),  
    },  
    as dag:  
  
    @task(task_id="get_names")  
    def get_names(**kwargs) -> list[str]:  
        ti: TaskInstance = kwargs["ti"]  
        ...
```



JSON Schema

Triggering a DAG via UI – Starting Simple

Fields detected by parameter type:

```
with DAG(  
    dag_id="summit",  
    params={  
        "some_text": "test-value",  
        "a_number": 42,  
        "a_bool": True,  
        "a_list": [  
            "one",  
            "two",  
            "three"  
        ],  
    },  
): ...
```

DAG conf Parameters

some_text:	<input type="text" value="test-value"/>
a_number:	<input type="text" value="42"/>
a_bool:	<input checked="" type="checkbox"/>
a_list:	<div>one two three</div>

Limitations: No Data/Type Validation, Labels, Descriptions, Required Check

Triggering a DAG via UI – Param Beautyness

Fields defined with **Param** object:

```
with DAG(  
    dag_id="summit",  
    params={  
        "a_number": Param(  
            42,  
            type="integer",  
            title="Your favorite number",  
            description="Everybody "  
                "should have a favorite number. What is your favorite?",  
        ),  
    },  
): ...
```

Benefits: JSON Schema Validation, Required Entry Check, Labels, Descriptions

DAG conf Parameters

	<input type="text" value="42"/>
Your favorite number *:	Everybody should have a favorite number. What is your favorite?

Triggering a DAG via UI – Many More Options

```
with DAG(  
    dag_id="summit",  
    params={  
        "select_a_item": Param(  
            "c",  
            type=["string", "null"],  
            title="Select something",  
            description_html="Select "  
            "a <code>cool</code> option(al).",  
            enum=["a", "b", "c"],  
            values_display={"a": "not good", "b": "better", "c": "one option"},  
        ),  
    },  
): ...
```

DAG conf Parameters

Select something:

one option



Select a **cool** option(al).

Benefits: JSON Schema Validation, Required Entry Check, Labels, Descriptions

Triggering a DAG via UI – Advanced Stuff

- `date`, `datetime`, `time`, `object` types
- `examples` for proposals to have free text bypassing the limits of `enum`
- `array` type for lists of strings or other `items` with JSON Schema types
- `section="Special advanced stuff..."` ...generates folded areas in the form
- `const` for hidden fields
- `custom_html_form` for all other stuff...

Take a look to `example_params_ui_tutorial` DAG and the [Params documentation](#)

DAG conf Parameters

Special advanced stuff with form fields ▾

Red:

Green:

Blue:

Pick a color *:

This is a special HTML widget as custom implementation in the DAG code. You can:

- Use sliders: To change RGB color.
- Change RGB HEX Code: Sliders will adjust.
- Additionally a preview color is shown.

Triggering a DAG via UI – Timeline & Outlook

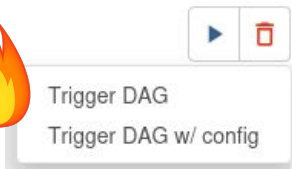
Airflow 2.6.0 – Initial Version

Airflow 2.6.1 – Fix rendering empty Array

Airflow 2.6.3 – Fix Number vs. Decimal

Airflow 2.7.0

- Trigger Button/Skip UI if no Param
- Multi-Select, Labels on Drop-downs, Proposed values, Non-String Arrays
- Bugfix JSON Propagation, Force Un-Pause



Airflow 2.7.1 – Bugfix Render “0” Default

Airflow 2.7.2 – Bugfix **None** values

Roadmap & Discussions

- [No Defaults in Required Fields](#) (31301)
- [Sync JSON <-> Form Fields](#) (31390)
- [Pre-Populate Fields via URL](#) (32859)
- [Handling of Empty Fields](#) (Null vs. Empty) (33923)
- More Field Types...?

➔ Looking forward for you contribution!

Questions?

Christian Schilling
Product Owner
christian.schilling@bosch.com



Jens Scheffler
Technical Architect
jens.scheffler@de.bosch.com

