CloudBlue
https://cloudblue.com

Dynamic Validation of Ordering Parameters

This article has been generated from the online version of the documentation and might be out of date. Please, make sure to always refer to the online version of the documentation for the up-to-date information.

Auto-generated at April 20, 2024

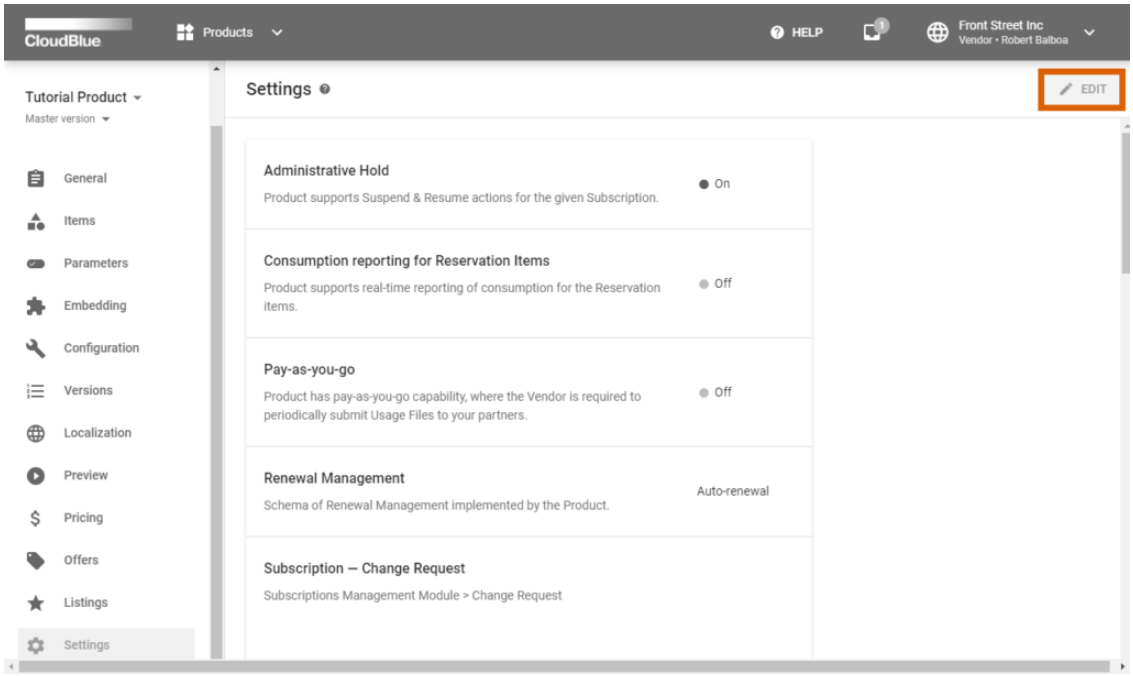
The following provides a use case and guidelines for configuring the dynamic validation of ordering parameters. It is highly recommended to familiarize yourself with the Dynamic Validation documentation that explains real-time validation concept and provides instructions on how to create and test out your webhook. Thereafter, follow the guidelines below to configure your ordering parameters dynamic validation.

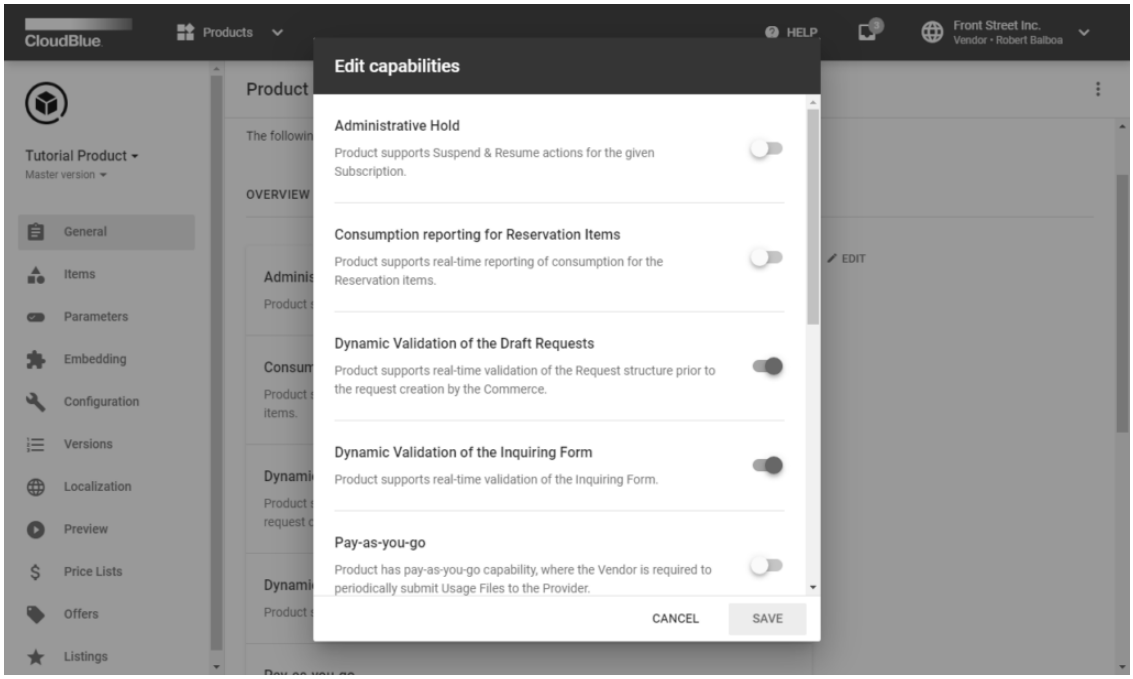
In addition, make sure that all of the following prerequisites are met:

- The Processor SDK Template is successfully deployed.
- Your deployed Processor is properly configured.
- Vendor API configuration is presented.

Enable Dynamic Validation

By default, the **Dynamic Validation of Draft Requests** capability for your specified product are disabled. Enable this capability as follows:





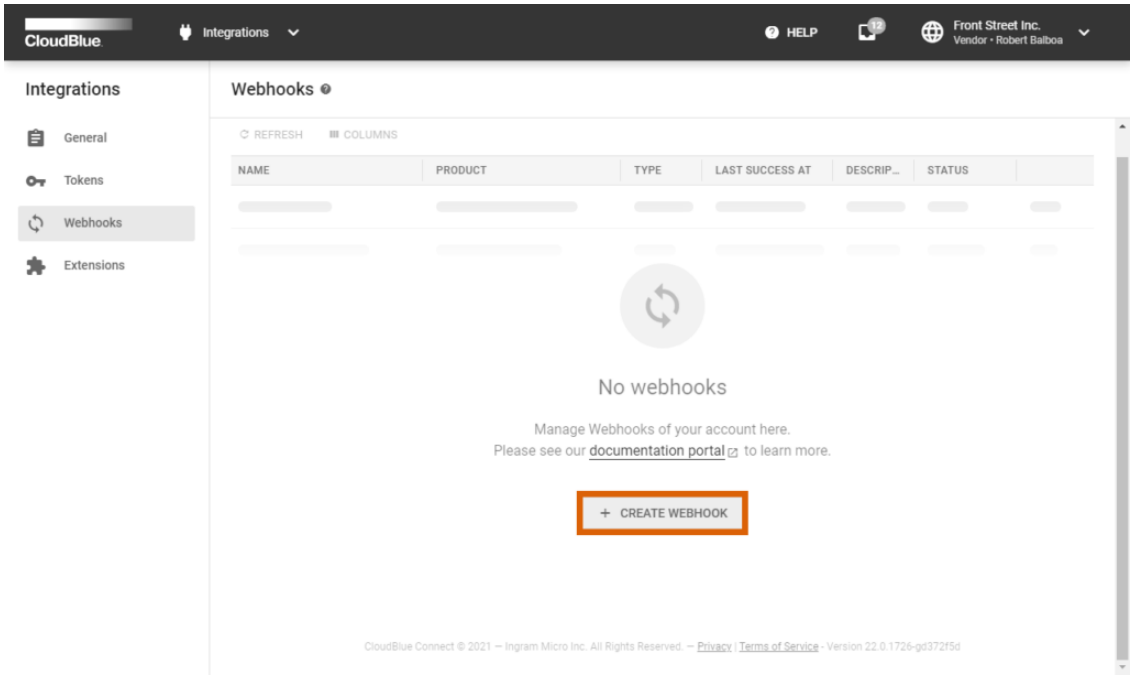
1. Navigate to the **Settings** section of your product profile page.
2. Click the **Edit** button to enable or disable your required capabilities.
3. Turn on required **Dynamic Validation** options and click **Save**.

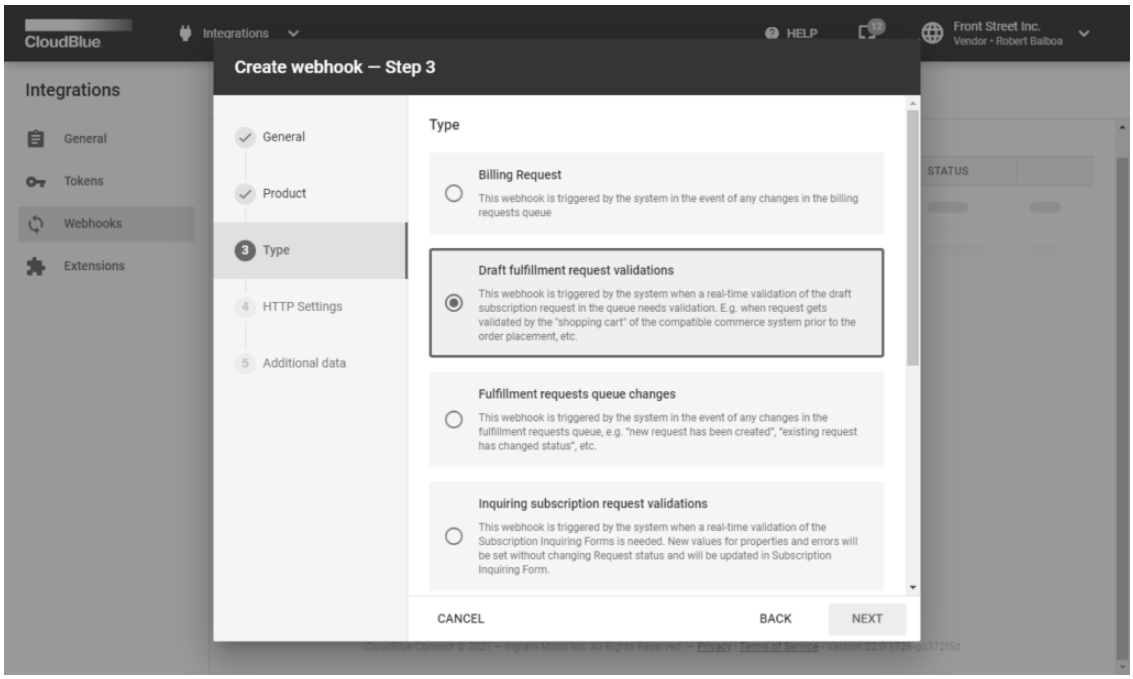
Therefore, your specified product will feature your assigned real-time validation capabilities. In case more information on provided option is required, refer to the Product Settings documentation.

Webhook Creation

Once the **Dynamic Validation of Draft Requests** capability is enabled, it is necessary to create your webhook on the Connect platform.

Access the **Integrations** module and navigate to the **Webhooks** section. Thereafter, click the **Create Webhook** button to launch a Webhook creation wizard.





Follow the wizard steps to configure your webhook. Make sure that the **Draft fulfillment request validations** type is selected.

For more information on how to successfully create and configure a webhook, refer to the Webhook documentation.

Code Snippets

Customize your real-time validation logic for your parameters within a created **dynamic_validation.py** file. State your expected value types and specify responses that trigger your provided error message.

```
from flask import Flask, request, json

api = Flask(__name__)

"""
# Configure FLASK to execute your script locally
set FLASK_APP=service
flask run
"""

def get_parameter_by_id(params, id):
    for param in params:
        if param['id'] == id:
            return param
    raise Exception('Parameter {id} not found.'.format(id=id))


def set_parameter(params, param):
    ret = []
    for p in params:
        if p['id'] == param['id']:
            ret.append(param)
        else:
            ret.append(p)
    return ret


def get_validation_request_data(request):
    data = request.data.decode("utf-8")
    jsondata = json.loads(data)
    return jsondata


# Your configured webhook calls the specified method to validate provided values within your
# ordering parameter during the subscription creation.
@api.route('/validate', methods=['POST', 'GET'])
def do_validate():

    jsondata = get_validation_request_data(request)
    params = jsondata['asset']['params']
```

```
    # Customize: Replace 'param_dynamic_validation' with the ID of the parameter that should be validated.
    param_1 = get_parameter_by_id(params, 'param_dynamic_validation')
    # Customize: Implement a required logic to validate provided values within the parameter.
    if param_1 and param_1['value'].isnumeric():
        return api.response_class(
            response=json.dumps(jsondata),
            status=200,
            mimetype='application/json'
        )

    else:
        # Customize: Provide an error message.
        param_1['value_error'] = "This error is from the validation script! Value should be numeric."
        params = set_parameter(params, param_1)
        jsondata['asset']['params'] = params
        response = json.dumps(jsondata)
        return api.response_class(
            response=response,
            status=400,
            mimetype='application/json'
        )
```

Run FLASK commands

Use the following FLASK commands to run your created **dynamic_validation.py** script:

```
set FLASK_ENV=debug

set FLASK_APP=dynamic_validation.py

flask run --port=5000 --host=127.0.0.1
```

The following provides an example that showcases how to utilize the provided commands:

```
(venv) D:\connect-processor-template-python-
boilerplate\{{cookiecutter.project_slug}}\connect_processor\app>set FLASK_ENV=debug
(venv) D:\connect-processor-template-python-
boilerplate\{{cookiecutter.project_slug}}\connect_processor\app>set FLASK_APP=dynamic_validation.py
(venv) D:\connect-processor-template-python-
boilerplate\{{cookiecutter.project_slug}}\connect_processor\app>flask run --port=5000 --host=127.0.0.1
```