https://cloudblue.com

Documentation Developer Resources Connect SDK and Tools

Connect Reports SDK

Auto-generated at May 3, 2026

Overview

Connect Reports SDK provides a set of tools and resources that enables Vendors and Distributors to independently generate custom reports on the CloudBlue Connect platform.

Custom reports allows utilizing any report parameters specified via Connect Report SDK. Therefore, custom reports can provide any required information for export. Additionally, Connect Reports SDK can be used to generate several custom reports at once.

Requirements

Python 3.6+
Make sure that you have a relevant environment to run the Python scripts. It is recommended to use **Python 3.6** or **above**.

Connect CLI SDK
Next, it is necessary to deploy Connect Command Line Interface. Refer to the Connect CLI GitHub project for more information.

Python OpenAPI Client
Install and deploy Connect Python OpenAPI Client. It is also recommended to familiarize yourself with the Python OpenAPI Client documentation.

Connect Reports Cookiecutter
Furthermore, deploy Cookiecutter for Connect Reports to bootstrap your project and to simplify your test operations and GitHub integrations.

Report Parameters

The following table introduces all available report parameter types, provides their examples and contains their overview. Place a required parameter within your **parameters** array from the **reports.json** file. Thereafter, code your specified parameter within the **entrypoint.py** file.

Type	Example	Overview
“product”	`{ "id":"product", "type":"product", "name":"Product list", "required":true, "description":"Select the products you want to include in report" },`	This parameter type represents your product list. Use the provided example to include specified products to your report.

“marketplace”	`{ "id":"mkp", "type":"marketplace", "name":"Marketplaces", "required":true, "description":"Select the marketplaces you want to include in report" },`	This parameter type defines your marketplaces. Include specified marketplaces to your report by using this example.
“hub”	`{ "id":"hub", "type":"hub", "name":"Hubs", "required":true, "description":"Select the hubs you want to include in report" },`	This parameter type represents your hub list. Include selected hubs by using the provided example.
“date_range”	`{ "id":"date", "type":"date_range", "name":"Report period", "description":"Provide the time period to create the report", "required": true }`	This parameter type is used to define your report date range. Specify your start date and end date by using this parameter type. Use the provided example to include and specify your report date range.
“date”	`{ "id":"date", "type":"date", "name":"date", "required":true, "description":"Select the date you want" },`	This parameter type is used to specify your report date. Include a date parameter by following the provided example.
“object”	`{ "id":"object", "type":"object", "name":"object", "required":true, "description":"give me a json object" },`	Use this parameter type to include a JSON object.
“single_line”	`{ "id":"single_line", "type":"single_line", "name":"Single line", "required":true, "description":"Tell me how you feel today in 1 sentence" },`	Include a single line by using this particular parameter type.

“checkbox”	`{`
	`"id":"rr_type",`
	`"type":"checkbox",`
	`"name":"Types of requests",`
	`"required":true,`
	`"description":"Select the type of requests you want to include in report",`
	`"choices":[`
	`{`
	`"value":"setup",`
	`"label":"Setup"`
	`},`
	`{`
	`"value":"update",`
	`"label":"Update"`
	`}`
		This parameter type allows utilizing checkboxes. Thus, by using these checkboxes, you can include specific information within your report. Define your checkboxes within the **choices** array.
		
		The provided example showcases how to include tier configuration requests and *setup*/*update* checkboxes to a report.

“choice”	`{`
	`"id":"rr_status",`
	`"type":"choice",`
	`"name":"Request status",`
	`"description":"Select the status of the requests you want to include in report",`
	`"required":true,`
	`"choices":[`
	`{`
	`"value":"tiers_setup",`
	`"label":"Tier Setup"`
	`},`
	`{`
	`"value":"inquiring",`
	`"label":"Inquiring"`
	`},`
	`{`
	`"value":"pending",`
	`"label":"Pending"`
	`},`
	`{`
	`"value":"approved",`
	`"label":"Approved"`
	`},`
	`{`
	`"value":"failed",`
	`"label":"Failed"`
	`}`
	`]`
	`}`
		Include specified information to your report and filter out unwanted data via this parameter type and specified *choices*. Define your filters within the **choices** array.
		
		The provided example demonstrates how to include fulfillment requests and their available statuses as choices to a report.

Report Parameters

Use Case

The following steps represent Connect Reports SDK usage scenario that showcases custom report creation workflow. A new custom report will be generated by completing the provided steps. This custom report will contain subscriptions (assets) of defined product for specified time period.

1. Prepare a project skeleton

Prepare your project skeleton via the corresponding Cookiecutter command. Make sure that all of the following files are presented:

- `__init__.py`
- `entrypoint.py`
- `readme.md`
- `template.xlsx`

Provide a report spreadsheet template within the **template.xlsx** file. Specify your custom report description in the **readme.md** file. Leave your other files untouched for now.

2. Edit the Report file

Access the provided **reports.json** file and specify your new custom report within this file.

```
{
   "name":"Custom_Project",
   "readme_file":"README.md",
   "version":"1.0.0",
   "language":"python",
   "reports":[
      {
         "name":"tutorial_report",
         "readme_file":"reports/tutorial_report/Readme.md",
         "template":"reports/tutorial_report/template.xlsx",
         "start_row":2,
         "start_col":1,
         "entrypoint":"reports.tutorial_report.entrypoint.generate",
         "audience":[
            "provider",
            "vendor"
         ],
         "report_spec":"1",
         "parameters":[
            {
               "id":"date",
               "type":"date_range",
               "name":"Report period",
               "description":"Provide the time period to create the report",
               "required":true
            },
            {
```

```
          "id":"product",
          "type":"product",
          "name":"Product List",
          "description":"Select the products you want to include in report",
          "required":true
        }
      ]
    }
  ]
}
```

- Enter your custom report *name* within the **reports** array. In this scenario, a custom report with “tutorial_report” name is created.
- Next, provide the path to your **readme** and **template** files. In this example, a custom report in the *tutorial_report* folder is created .
- Specify start row and start column for your report via the `"start_row"` and `"start_col"` attributes.
- Provide your *entrypoint* as in the following example: `"entrypoint":"reports.{your_report_name}.entrypoint.generate"`
- Define your Vendor and/or Distributor (Provider) audience for your report within the following **audience** array. In this scenario, both Distributors and Vendors are selected.
- Finally, place required report parameters in the **parameters** array. In this example, the *date range* and *product* parameters are presented.

3. Edit the Entrypoint file

Open your created **entrypoint.py** file and specify your **report generate** code within this file. Use the RQL client or string concatenation to successfully create an entrypoint file. In this example, the RQL client is used.

```
import cnct import R
def generate(client, parameters, progress_callback)
  rql = R().events.created.at.ge(parameters['date']['after'])
  rql &= R().events.created.at.le(parameters['date']['before'])
  rql &= R().assets.product.id.oneof(parameters['product'])
  requests = client.requests.filter(rql).all()
  total = requests.count()
  counter = 0
  for requests in requests:
    yield(
      request['id'],
      request['assets']['id']
    )
    counter +=1
    progress_callback(counter,total)
```

- **Client**: This client represents Connect OpenApi client instantiated with a valid token. Thus, it allows accessing the account from where the report is triggered. Refer to the Python OpenAPI Client documentation for more details.
- **Parameters**: The parameters contain your dictionary with all specified parameters and parameter values. In this example, *date after/before* and *product* parameters are presented.
- **Progress_Callback**: Define the progress callback via this function. This function provides a mechanism to inform (either via Connect Portals or CLI) about the progress of your report generation.

4. Generate your report

Once all required information is provided and your code is ready, run the following command via the Connect CLI:

```
$ ccli report execute tutorial_report -d ./tutorial_report
```

Thus, the tutorial report will be successfully generated. Furthermore, report results will be available via a generated **XLSX** file.