

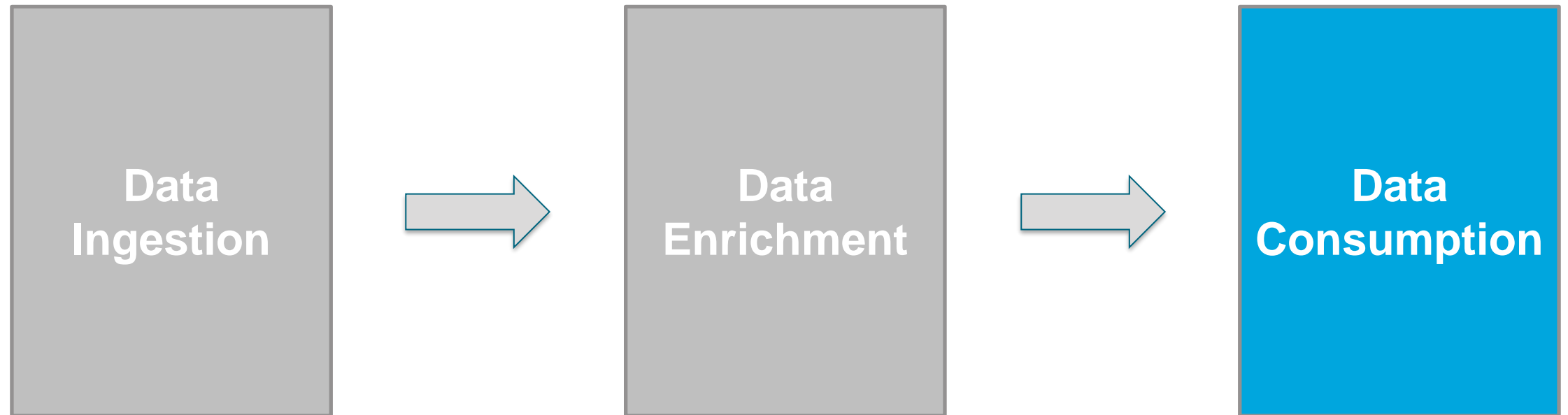


# *OSDU Data Platform Application Developer Training*

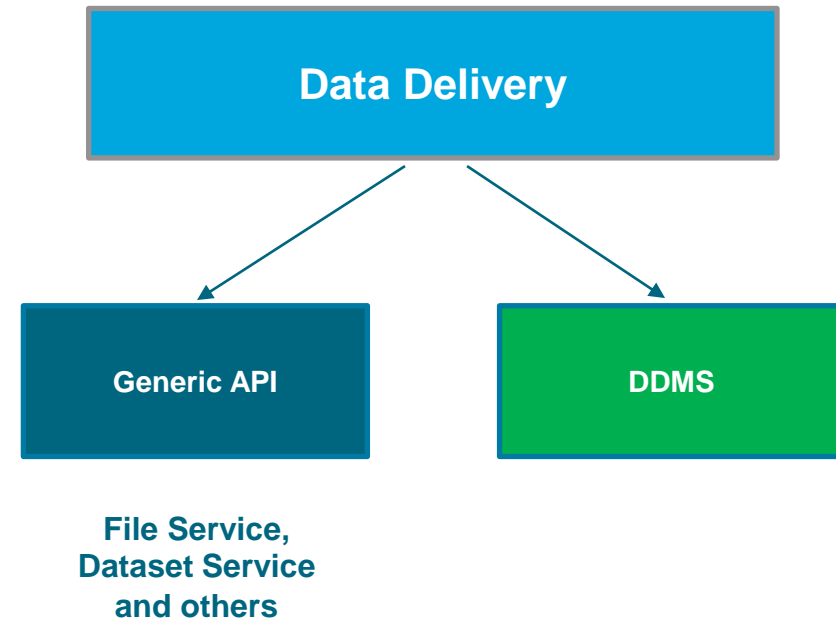
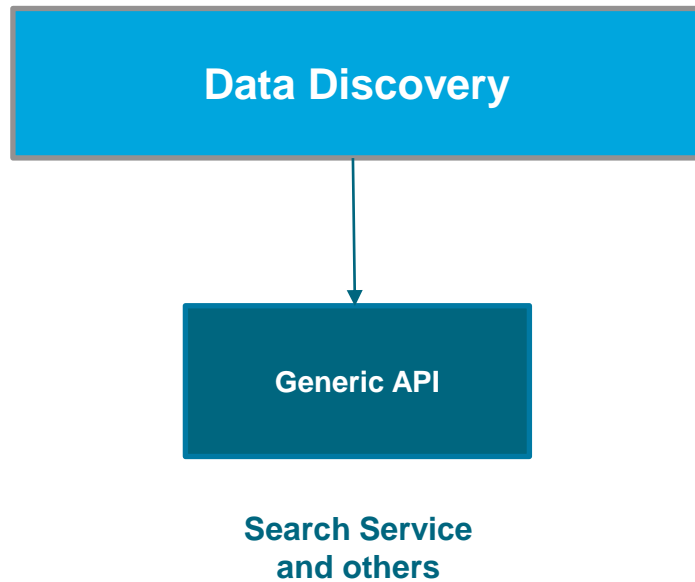
## *Data Life Cycle Lab*



# Data Life Cycle



# Data Consumption



# End-to-end Application

- » These training materials are based on the Quickstart application
- » You can explore the source code and run the application
  
- » Backend in Python
- » Frontend in React

# Authorization: Configuration

- » Register your application with the Identity Provider.
- » Get **client\_id**, **client\_secret** and **auth\_discovery\_url** from your OSDU administrator or cloud provider.

```
1  OSDU_AUTH_CLIENT_ID="<your_client_id>"
2  OSDU_AUTH_CLIENT_SECRET="<well_hidden_secret>"
3  OSDU_AUTH_DISCOVERY_URL="<cloud_specific_auth_discovery_url_2.0>"
4  OSDU_AUTH_REDIRECT_URL="http://localhost:8080/auth/callback"
5  OSDU_AUTH_SCOPES="openid profile offline_access <your_client_id>/default"
6  OSDU_DATA_PARTITION="opendes"
```

- » Make sure **redirect\_url** is whitelisted in cloud settings.

# Authorization: Client initialization

```
1  from oic.oic import Client
2  from oic.utils.authn.client import CLIENT_AUTHN_METHOD
3
4  oidc_client = Client(client_authn_method=CLIENT_AUTHN_METHOD, verify_ssl=False)
5  oidc_client.provider_config(issuer=OSDU_AUTH_DISCOVERY_URL)
6
7  registration_info = RegistrationResponse(
8      |   client_id=OSDU_AUTH_CLIENT_ID,
9      |   client_secret=OSDU_AUTH_CLIENT_SECRET,
10 )
11 oidc_client.store_registration_info(registration_info)
```



# Authorization: Redirect to authorization endpoint

```
1  # scopes depend on the cloud provider
2  scopes = []
3  for item in OSDU_AUTH_SCOPES.split(","):
4      |   scopes.append(item)
5
6  # set parameters for authorization server request
7  args = {
8      |   "response_type": "code",
9      |   "client_id": oidc_client.client_id,
10     |   "scope": scopes,
11     |   "redirect_uri": OSDU_AUTH_REDIRECT_URL,
12     |   "state": session.state,
13 }
14
15 # create authorization request
16 auth_request = oidc_client.construct_AuthorizationRequest(request_args=args)
17
18 # get login url from provider configuration
19 login_url = auth_request.request(oidc_client.authorization_endpoint)
20 raise HTTPFound(login_url) # HTTP 302
```

# Authorization: Get tokens

```
1  callback_response = oidc_client.parse_response(  
2  |  AuthorizationResponse,  
3  |  info=request.query_string,  
4  |  sformat="urlencoded")  
5  
6  # extract authorization code from auth server response  
7  args = {  
8  |  "code": auth_response["code"],  
9  |  }  
10  
11 # exchange authorization token for the access token  
12 access_token_response = oidc_client.do_access_token_request(  
13 |  state=auth_response["state"],  
14 |  request_args=args  
15 |  )  
16  
17 # extract tokens  
18 access_token = access_token_response["access_token"]  
19 refresh_token = access_token_response["refresh_token"]
```

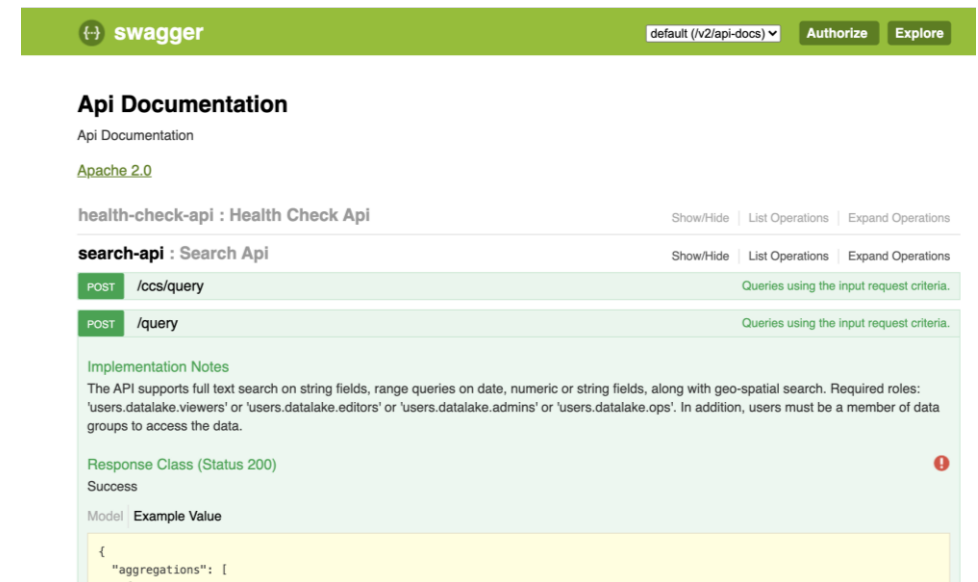


# Authorization: Assemble headers

```
1 # Headers to be attached to every API call
2 headers = {
3     "data-partition-id": OSDU_DATA_PARTITION,
4     "Authorization": f"Bearer {self.access_token}",
5 }
```

# Explore service API

- » General list of services is well documented at <https://community.opengroup.org/osdu/documentation/-/wikis/Core-Services-Overview>
- » Every service has its own swagger page
  - `<osdu_endpoint>/api/search/v2/swagger-ui.html`
  - `<osdu_endpoint>/api/file/swagger-ui.html`



The screenshot shows the Swagger UI for the search-api. At the top, there is a green header with the Swagger logo, a dropdown menu set to 'default (/v2/api-docs)', and buttons for 'Authorize' and 'Explore'. Below the header, the page is titled 'Api Documentation' and includes a link to 'Apache 2.0'. The main content area lists two APIs: 'health-check-api : Health Check Api' and 'search-api : Search Api'. The 'search-api' is expanded to show a 'POST /query' endpoint. The description for this endpoint states: 'Queries using the input request criteria.' Below the description, there are sections for 'Implementation Notes', 'Response Class (Status 200)', and 'Success'. The 'Implementation Notes' section contains the text: 'The API supports full text search on string fields, range queries on date, numeric or string fields, along with geo-spatial search. Required roles: 'users.datalake.viewers' or 'users.datalake.editors' or 'users.datalake.admins' or 'users.datalake.ops'. In addition, users must be a member of data groups to access the data.' The 'Response Class (Status 200)' section shows a 'Success' status. The 'Model' section is titled 'Example Value' and displays a JSON snippet: 

```
{
  "aggregations": [
```

# Two search options

- » Full-text
- » Geo-spatial

The screenshot shows a web application interface with a blue header bar. On the left, there is a search input field containing the text 'BIR' and a magnifying glass icon, followed by a 'SEARCH' button. Below the header bar is a teal bar with the text 'Search on map'. The main content area is white and contains the text 'Results will be displayed here'. On the right side of the interface, there are two buttons: 'LOGIN' and 'LOGOUT'. Below these buttons, the text 'No Trajectory to display' is shown in bold, followed by the instruction 'Find well and click visualize to appropriate data set'.

# Search Service: Make a request

- » get `osdu_instance_url` from your OSDU administrator or cloud provider

```
79  OSDU_API_SEARCH_URL="https://<osdu_instance_url>/api/search/v2/query"
80
81  search_response = requests.post(
82      |         OSDU_API_SEARCH_URL,
83      |         headers=headers,
84      |         json=search_query)
85
86  search_response_json = search_response.json()
```

# Search Request: Construct a query

```
66 # get well name from request, such as "A05-01"
67 well_name = request.params.get("well_name")
68
69 search_query = {
70     "kind": "opendes:wks:master-data--Well:1.0.0",
71     "query": f"data.FacilityID: \"{well_name}\"",
72     "returnedFields": [
73         "data.FacilityID",
74         "id",
75         "data.SpatialLocation.Wgs84Coordinates.geometries"
76     ]
77 }
```



Q A05-01 SEARCH

# Search Request: View response data

```
1  {
2    "results": [
3      {
4        "data": {
5          "FacilityID": "A05-01",
6          "SpatialLocation.Wgs84Coordinates": {
7            "geometries": [
8              {
9                "coordinates": [
10               3.51906683,
11               55.68101428
12             ],
13             "type": "point"
14           }
15         ]
16       }
17     },
18     "id": "opendes:master-data--Well:8438"
19   }
20 ],
21 "totalCount": 1
22 }
```

# Spatial Search: Construct a query

```
1 # create Search API request payload:
2 search_query = {
3     "kind": "opendes:wks:master-data--Well:1.0.0",
4     "spatialFilter": {
5         "field": "data.SpatialLocation.Wgs84Coordinates",
6         "byGeoPolygon": {
7             "points": [
8                 {
9                     "longitude": 5.1580810546875,
10                    "latitude": 52.859180945520826
11                },
12                ...
13            ]
14        },
15    },
16    "returnedFields": [
17        "data.FacilityID",
18        "id",
19        "data.SpatialLocation.Wgs84Coordinates.geometries"
20    ],
21 }
```





# Search Request: Query related records

```
97 well_id = request.params.get("well_id")
98
99 search_query = {
100     "kind": "opendes:wks:master-data--Wellbore:1.0.0",
101     "query": f"data.WellID: \"{well_id}\"",
102     "returnedFields": [
103         "id",
104         "data.NameAliases.AliasName"
105     ]
106 }
107
108 search_response = requests.post(
109     OSDU_API_SEARCH_URL,
110     headers=headers,
111     json=search_query)
```

# Fetch data: Signed URL and file download

```
124 file_id = ids_list[0]
125
126 ✓ file_response = requests.get(
127     f"{OSDU_API_FILE_URL}/{file_id}/downloadURL",
128     headers=headers
129 )
130
131 signed_url = file_response.json()["SignedUrl"]
132
133 blob_storage_response = requests.get(signed_url)
134 file_content = blob_storage_response.content
```

```
1 ✓ {
2     "SignedUrl": "https://[REDACTED]/file-persistent-area/
   osdu-user%2Fopen-test-data%2FR3%2Fsource_files%2FTNO_Source_Files%2Ftrajectories%2F5496.csv?sv
   sig=F7P%2F6XcYZdr%2Bcg11JESc0yMckRLX36HgmeGE%2BgH%2Bwms%3D"
3 }
```

# Fetch data: Obtain dataset ids

```
113 search_query = {
114     "kind": "opendes:wks:work-product-component--WellboreTrajectory:1.0.0",
115     "query": f"data.WellboreID: \"{wellbore_id}\"",
116     "returnedFields": [
117         "data.Datasets"
118     ]
119 }
120
121 search_results = call_search_api(request, search_query)
122
123 ids_list = [record["data"]["Datasets"] for record in search_results["results"]]
```

```
1 {
2   "results": [
3     {
4       "data": {
5         "Datasets": [
6           "opendes:dataset--File.Generic:e1a9aaf71a2442e59db0dada8be7d86d:"
```

# Application Development Guide

- » [https://community.opengroup.org/osdu/tutorials/guides/-/blob/master/OSDU\\_Release2\\_Application\\_Development\\_Guide.pdf](https://community.opengroup.org/osdu/tutorials/guides/-/blob/master/OSDU_Release2_Application_Development_Guide.pdf)
- » Comprehensive and structured information about application development for OSDU platform
- » Written for R2 but most of the information is relevant
- » This document is available to everyone as our contribution to OSDU