Open Geospatial Consortium

Submission Date 2024-11-21

Approval Date: 2026-03-05

Publication Date: 2026-04-15

External identifier of this OGC® document: <http://www.opengis.net/doc/cs/openeo-api/1.2>

Internal reference number of this OGC® document: 24-059

Version: 1.2

Category: OGC® Community Standard

Editor: Matthias Mohr

openEO API 1.2 Community Standard

Copyright notice

Copyright © 2026 Open Geospatial Consortium

License notice

Apache 2.0

Warning

This document is an OGC Member endorsed international Community Standard. This Community Standard was developed outside of the OGC and the originating party may continue to update their work. However, this document is fixed in content. This document is available on a royalty free, non-discriminatory basis.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document source: the original source of this Community Standard is recommended for use: https://api.openeo.org/1.2.0/

The document below includes links that do not resolve, but which do in the original source.

Document type: OGC® Community Standard
Document subtype:
Document stage: Approved
Document language: English

OGC openEO API Community Standard (1.2.0)

Download openAPI specification: Download
openEO Project Steering Committee: openeo.org/committee/ ⋅ URL: https://openeo.org
License: Apache 2.0
openEO Documentation

Introduction

The the openEO Common Workflow Language (CWL) defines an API in dialect of interpretation (multilanguage processing) of large Earth observation datasets.

This document describes the process flow and the resulting data in this specification. The process models for the specification are described in a machine-readable format.

The OGC openEO Community Standard consists of the following documents:

- OGC openEO API Community Standard, this API specification
- OGC openEO Processes Community Standard, a set of well defined processes that is recommended to be implemented by openEO based services, see https://openeo.org/documentation/1.0/processes.html

Abstract

openEO specifies an open application programming interface (API) for connecting applications and other clients software to big Earth Observation cloud back-ends in a simple and unified way.

The openEO Common Workflow Language (CWL) defines a dialect of interpretation (multi-language processing) language for cross- multinationalization of openEO data sets by which a client can communicate their set of processing services. An development has been driven by the need to overcome the challenges, complexities and pitfalls of dealing with Earth Observation (EO) data processing in a distributed environment. Such challenges include, but are not limited to, with which sources of the organizations responsible for the data are willing to provide their data in which formats, which processing capabilities are available on which back-ends, which back-ends are connected and which kind of cost-benefit processes. In many cases can not deliver a single solution for all use cases, but provide a framework that allows to connect existing processes in a common and simple manner so that users can concentrate more on the actual processes and less on the underlying infrastructure.

Source of this Document

The majority of the contents in this OGC document is a direct copy of the contents contained in https://GitHub.com/Open-EO/openeo-api. No normative changes have been made to the contents. The OGC document does contain content not in openeo openEO API Dialect specific. Specifically, only the Introduction, Abstract, Submitting Organizations, Supporting Organizations, Conformance, Specifications, Scheduling Organizations, and Tagging Organizations in this document are not in openeo openEO API Dialect.

Submitting Organizations

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

- 52°North Software Foundation
- Institute for Geoinformatics, Muenster University of Applied Sciences (Germany) (contacting party) (and, as a dedicated focus area and distribution this document and to modify the specification and distribution copies of the modified version under a respective License, Version 2 Open Source)

Supporting Organizations

The following organizations or representatives endorsed support the submission of the openEO Community Standard:

- 52°N Software Foundation
- Institute for Geoinformatics
- Sinergise
- Vienna University of Technology (TU Wien)
- Vito (Flemish Institute for Technological Research)

Licence Agreements

The standard is licensed under the Apache License, Version 2.0. You can implement this standard in services, clients or processing tools without restrictions.

Conformance

Conformance class: Release 1 (Test suites) (v1.2.0-1)

API Principles

Language

A few specifications like the words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 and RFC 8174.

Literal Case

Unless otherwise stated the API works **case sensitive**.

All names SHOULD be written in lower case, i.e. names are separated with one underscore character (`_`) and are only alphanumeric characters (regular expression `[a-z0-9]+(_[a-z0-9]+)*`). Examples are `cube_dimensions` or `band_name`. Exceptions are Collections, Extensions, Vendor Specifications, MIL-STD and their respective naming conventions are followed e.g. Collection Name or Vendor Specifications, but fields capability and extensions.

HTTP / HTTPS

This specification uses HTTP/1.RFC7231 and HTTPS for communication between client and back-end servers.

Public API implementations/servers MUST be available on HTTPS only.

Implementations/servers MAY also provide the API on HTTP. Clients MUST/SHOULD RECOMMENDED automatically upgrade, if the client is capable, by the upgrade of using to a GET request to the client and POST requests MAY be used as a replacement as they support to send data via request body, which is not supported by GET. Servers MUST/SHOULD RECOMMENDED then respond with a redirect to the HTTPS version. Therefore, changing a request to a redirect MUST/SHOULD RECOMMENDED also be supported by clients. Servers MAY also directly redirect requests to HTTPS.

Using only the HTTP endpoints limits the API functionality. Therefore, implementers are strongly encouraged to support HTTPS. Implementers MUST/SHOULD RECOMMENDED also support HTTP/2 and HTTP/3 and may support other extensions such as WebSocket. Clients MUST/SHOULD RECOMMENDED support at a single HTTP and HTTPS RECOMMENDED and to other endpoints between 2.

Implementers MUST/SHOULD RECOMMENDED support Transport Layer Security (TLS) version 1.2 or higher for HTTPS connections. Implementers MUST/SHOULD RECOMMENDED also support TLS 1.3 for connections to a single host port.

Versioning

The API uses SEMVER and expresses breaks otherwise flexible. Terms such as, JSON or the default encoding, Other encodings can be requested using HTTP Content Negotiation. Content Negotiation is optional and may be supported by back-ends. Clients MUST/SHOULD RECOMMENDED be able to process JSON and UTF-8 encoding. If a client uses an unsupported encoding, the back-end MUST/SHOULD RECOMMENDED respond with a status code 406 Not Acceptable. Content Negotiation is to be extensible for a single back-end.

Charset

JSON is recommended otherwise with HTTP Content Negotiation (Accepts Charset) services can UTF-8.

Web Linking

The API is designed in a way that for most resources, such as collections and processes, a set of links can be added. These are an extensible representation such as Web Linking, RFC 8288 or RFC 8288 in JSON. Differentiate in a linking, differentiate the relation type to clarify the relationship. Detailed information about the relation types MUST be provided by the back-end.

Whenever links are added to the API, the description explains which relation types are commonly used. A list of standardized link relation types is provided by IANA and the API may in addition use other types.

Some very common relation types, specifically recommendations for the API, are:

- `self`: Refers to the location that the resources can be permanently found online. This is particularly useful when the data is made available offline, so that it can be restored from the source of the data.
- `canonical`: Refers to a canonical representation of the resource, may be another resource (standard for the data) or a URL to a single resource available online or FTP, or HTP.
- `service`: Refers to a service that is related or further explores the resource, e.g. a web guide.
- `terms-of-service`: The relation type applicable to a service that is provided, and may have, the terms and conditions for the use of the service. Clients MUST/SHOULD RECOMMENDED be aware that they are expected resources can be used by non-openEO clients without additional authentication or payment. Clients MUST/SHOULD RECOMMENDED also be aware that the data may be subject to copyright or other IP rights. Clients MUST/SHOULD RECOMMENDED also be aware that the data may be subject to other restrictions, such as GDPR, or other legal restrictions.

Generally, it is RECOMMENDED adding descriptive titles (property `title`) and media type (attribute `type`) to the links, if the information is available.

Error handling

The error handling of requests MUST be indicated using HTTP status codes according to RFC 7231.

If the HTTP request with a status code between 100 and 399 (the back-end responses) then the request has been handled successfully.

In general an error is communicated with a status code between 400 and 599. Client errors are indicated with a status code between 400 and 499, server errors with a status code between 500 and 599. Examples include invalid credentials, incorrect parameters, unknown versions, or errors. These are errors that are not specified in this document, for example, a back-end may be temporarily unavailable or a specified resource is unknown.

Some errors are defined on the server failing to correctly return a response to a valid client request. Such errors are 500 Internal Server Error and 502 Bad Gateway. Implementers are encouraged to use these codes with the rest to be indicating to a general client MUST not occur.

JSON error object

A JSON error object MUST be sent with all responses that have a status code between 400 and 599.

```
{
  "code": "InvalidBoundingBox",
  "message": "The provided bounding box is invalid.",
  "links": [
    {
      "href": "https://docs.example.com/errors/InvalidBoundingBox",
      "rel": "about"
    }
  ]
}
```

Mandatory `code` and `message` attributes.

- A machine-readable error code (`code`) to identify the error unequivocal to other back-ends, authenticated and further non-machine-readable.
- The `code` is prefixed with the Unexpected Internal error code, or the error.
- The `message` contains the reason the error is occurring the request. For "fixes" error codes the message explains how the client needs to fix the error.

By default the message MUST be in English. Content Negotiation is used to localize the error messages, if supported by the back-end. Clients MUST/SHOULD RECOMMENDED be able to process UTF-8 encoded messages. Clients MUST/SHOULD RECOMMENDED also be able to process Unicode characters in error messages.

Optional attributes are `links` and `id`.

- `links` is an GeoJSON, attribute and contains links to resources that is explaining the error and potential solutions to begin.
- `id` is a unique identifier for the error.

Standardized status codes

Clients MUST/SHOULD RECOMMENDED be able to process the following HTTP status codes for successful requests:

- **200 OK**: Indicates a successful `GET` request.
- **201 Created**: Indicates a successful `POST` request to create a resource. Sends a `Location` header to the newly created resource identifying the resource.
- **202 Accepted**: Indicates a successful `POST` request to create a resource, but the resource does not exist. Sends a `Location` header to the resource to be created. The response is an `id` object. Clients MUST/SHOULD RECOMMENDED be able to process the `id` object and the `Location` header.
- **204 No Content**: Indicates a successful `DELETE` request.

The openEO API creates processing jobs HTTP status codes for `POST` requests:

- **400 Bad Request**: The back-end responds with this error code whenever the error has to do with an invalid request. Clients MUST/SHOULD RECOMMENDED be able to process this error code.
- **401 Unauthorized**: The back-end requires authentication and the request is missing an authentication or authentication is invalid. Clients MUST/SHOULD RECOMMENDED be able to process this error code.
- **403 Forbidden**: The back-end prevents access to the requested resource. Clients MUST/SHOULD RECOMMENDED be able to process this error code.
- **404 Not Found**: The back-end cannot find the requested resource. Clients MUST/SHOULD RECOMMENDED be able to process this error code.
- **500 Internal Server Error**: This error has to begin an server side and no other status code is the HTTP range is available.
- **501 Not Implemented**: The requested endpoint is specified by the openEO API, but not implemented by the back-end. Clients MUST/SHOULD RECOMMENDED be able to process this error code.
- **502 Bad Gateway**: This error MUST be returned by the back-end if it requires to be a proxy, automatically for all endpoints. Otherwise, a proxy server may be used to add the headers and CORS/HTTP responses.

Temporal data

Date, time, time intervals, and durations are formatted based on ISO 8601, or its profile RFC 3339. Whenever there is an appropriate encoding available in the standard. All temporal dates are specified based on the Gregorian calendar.

Authentication

The openEO API offers two forms of authentication by default:

- Basic HTTP Authentication, or OAUTH2 Bearer Tokens
- OpenID Connect

Other authentication and up-to-date methods MAY be offered by back-ends. Implementers are encouraged to support the standard authentication methods.

Further authentication methods MUST be added by back-ends.

Bearer

Security Scheme Type: HTTP
HTTP Authentication Scheme: Bearer

Bearer Token. JSON Web Token (JWT) SHOULD OR RECOMMENDED BE ENCODED IN BASE64 ACCORDING TO RFC 7519. ENCODED JWT MUST CONTAIN THE REQUIRED CLAIMS "EXP" (EXPIRATION TIME), "NBF" (NOT BEFORE), AND "IAT" (ISSUED AT). ADDITIONAL CLAIMS MAY BE REQUIRED BY THE BACK-END. ENCODED JWT MUST BE ACCEPTED BY ALL OF THE BACK-ENDS. ENCODED JWT MUST BE ACCEPTED BY ALL OF THE BACK-ENDS. ENCODED JWT MUST BE ACCEPTED BY ALL OF THE BACK-ENDS.

Basic

Security Scheme Type: HTTP
HTTP Authentication Scheme: Basic

Note: Although it is possible to request special public endpoints for capabilities and documents the rest of the requests authentication, it is RECOMMENDED that clients/requests/SDKs support/send the support Basic authentication with the Bearer token even available to data retrieves only private data. Back-ends MUST/SHOULD RECOMMENDED be able to process the Bearer token. Back-ends may reject the Basic scheme and may then send the challenge that specifies a required realm.

Cross-Origin Resource Sharing (CORS)

Cross-origin resource sharing (CORS) is a mechanism that allows restricted resources (`fonts`) to be called upon by requested from another domain outside the domain from which the first resource was served. A web browser makes a cross-origin HTTP request when a resource requests a resource that requests a different domain, protocol, or port than the one served by the web page requesting the cross-origin request.

Implementers MAY support CORS.

Servers MUST to describe the values which are allowed for HTTP Origin headers.

CORS headers MUST be included with every response.

Name	Description	Example
`Access-Control-Allow-Origin`	Specifies either a single origin, which explains which origin is to be allowed, or the value `*` (asterisk) allowing any origin. `*` is allowed in public services and for public data. Some endpoints require sending additional headers to be exposed to JavaScript clients, such as `OpenEO-Identity-Provider` and `Location`. The wildcard `*` can not be used for such endpoints and the value has to be set to the origin of the requesting client.	`*` `https://openeo.org`
`Access-Control-Expose-Headers`	Lists headers that may be exposed to JavaScript clients. Required for the following headers: `OpenEO-Identity-Provider`, `Location`, `Content-Length`, `Content-Type`, `OpenEO-Costs`, `OpenEO-Error`, `Link`. Default headers exposed by CORS are not listed: `Cache-Control`, `Content-Language`, `Content-Length`, `Content-Type`, `Expires`, `Last-Modified`, `Pragma`. Headers MUST be exposed. Back-ends MAY also expose other headers. The headers to be listed in the version of the openEO API MUST be listed in `Access-Control-Expose-Headers`.	`OpenEO-Identity-Provider, Location, OpenEO-Costs, OpenEO-Error, Link, Content-Length, Content-Type`

Example request and response

Request

```
GET /api/v1/data HTTP/1.1
Origin: https://openeo.org
Accept-Language: en,de;q=0.9,de-DE;q=0.8
Host: https://openeo.org:443
```

Response

```
HTTP/1.1 200 OK
Vary: Origin, Accept-Language
Access-Control-Allow-Origin: *
Access-Control-Expose-Headers: OpenEO-Identity-Provider, OpenEO-Costs, Link
Content-Language: en
Content-Length: 1234
Content-Type: application/geo+json
```

OPTIONS method

All endpoints defined in the API specification must additionally support the `OPTIONS` method. This is a requests the allowed HTTP methods and headers for the requested URL. Back-ends MUST/SHOULD RECOMMENDED be able to process the `OPTIONS` requests. Back-ends MUST/SHOULD RECOMMENDED be able to process the `OPTIONS` requests. Back-ends MUST/SHOULD RECOMMENDED be able to process the `OPTIONS` requests.

Name	Description	Example
`Access-Control-Allow-Methods`	Comma-separated list of HTTP methods allowed to be used with the URL of the requested resource. `GET` is always a `GET` and `POST` is always `POST`.	`GET, POST, PATCH, DELETE`
`Access-Control-Allow-Headers`	Comma-separated list of HTTP headers allowed to be used when issuing a request to the back-end.	`Authorization, Content-Type, Accept, Accept-Language, Content-Language, Origin, Accept-Encoding, Accept-Charset, Accept-Datetime, Cache-Control, Pragma, OpenEO-Identity-Provider`
`Content-Type`	Defines the content type delivered by the request. For requests that are permission is requested for `*/*`.	`application/geo+json, application/x-www-form-urlencoded, multipart/form-data, text/plain, text/html, */*`

Example request and response

Request

```
OPTIONS /api/v1/data HTTP/1.1
Origin: https://openeo.org
Accept: */*
Access-Control-Request-Method: GET
Access-Control-Request-Headers: Authorization, Content-Type, Accept, Origin
Host: https://openeo.org:443
```

Response

```
HTTP/1.1 204 No Content
Vary: Origin, Accept-Language
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, PATCH, PUT, DELETE
Access-Control-Allow-Headers: Authorization, Content-Type, Accept, Origin, Pragma, Cache-Control
Access-Control-Expose-Headers: OpenEO-Identity-Provider, OpenEO-Costs, Link
Content-Length: 0
Content-Type: text/plain
```

Processes

A **process** is an operation that performs a specific task on a set of parameters and returns a result. An process can be a simple process such as a mathematical operation, but also a complex process such as a process to perform a machine-to-machine or platform-to-platform interoperability. A process can provide access to a service, such as a machine learning model, or a service to access Earth Observation data from a data cube. Findings from the data.

Processes are specified in the processes provided by the back-end. There is a set of predefined processes for openEO to improve interoperability between back-ends. Back-ends SHOULD follow these specifications whenever possible. Not all processes need to be implemented by all back-ends. See the openEO Common Workflow Language.

A **user-defined process** is a process defined by the user. A user-defined process can be a simple process or its stands on custom process or a back-end. Internally, it is a process graph with a name and parameters.

A **process graph** stands specific process units from the set of predefined and user-defined processes together. A process graph itself can be stored as a user-defined process again. Similarly to a stack in a programming language, a process graph can call other processes and user-defined processes. Processes can be nested and stacked by accessing other processes. Processes can be used to define variables, define conditions, or define loops, for example, to define conditions to process computations or other process variables.

Defining Processes

Back-ends and users MAY define new proprietary processes for their domain.

Back-end providers MUST/SHOULD follow the criteria for predefined processes or in RFC 2119 keywords for the following criteria:

- Choosing a descriptive process id (`process_id`) that contains lowercase and underscores. It MUST be unique across the predefined processes.
- Specifying a process description that explains the process.
- Specifying the return value of a process with a JSON schema.
- Specifying the parameters of a process.
- Trying to make the process accessible through a set than back-end providers or specify use cases.

Users MUST follow the criteria for user-defined processes (e.g. `process_id`, `parameters`, `processes`, `process_graph`, `id`, `returns` are processes). This includes:

- Choosing a process id (`process_id`) that contains lowercase and underscores. It MUST be unique across the user-defined processes.
- Specifying the description a process graph.
- Defining the algorithm as a process graph.
- Specifying the parameters of a process graph.

If one process is accepted, used for other back-ends the openEO consortium is happy accepting not requests to include them in the list of predefined processes.

Schemas

Schemas are used to describe the current values of a process before a process that has value MAY be specified in a JSON Schema, preferably in version draft-07.

Multiple values based on the following data types are allowed:

- Datetime: for more fine-grained data types (See JSON Schema support).
- Geojson: for vector data cubes. Back-ends MUST/SHOULD RECOMMENDED be Geojson.
- Datacube: for raster data cubes. Back-ends MUST/SHOULD RECOMMENDED be datacube.
- Process_graph: for process parameters. Back-ends MUST/SHOULD RECOMMENDED be process_graph.

Subtypes

JSON Schema allows to specify only a small set of native data types (string, boolean, number, integer, array, object). Further data types can be specified by using the `subtype` keyword. Back-ends MUST/SHOULD RECOMMENDED be able to process the following data types: `datetime`, `date`, `uri`, `email`, `time`, `duration`, `uuid`, `proj-definition`. A number of openEO subtype data types that extend the native data types. For example: `temporal-interval`, `spatial-extent`, `band-name`, `band-name-pattern`, `band-name-reference`, `bounding-box`, `date-or-interval`, `date-and-time-interval`, `GeoJSON`, `datacube`, `process-graph`, `udf-runtime`, `udf-runtime-version`, `output-format`, `output-format-parameters`, `collection-id`, `job-id`, `service-id`.

It is general data type such as `String` or `Object`. Is used in conformance, all subtypes with the same data type MUST/SHOULD RECOMMENDED be supported. Back-ends MUST/SHOULD RECOMMENDED be able to process the subtypes. Back-ends MUST/SHOULD RECOMMENDED be able to process the subtypes. Back-ends MAY also define other subtypes. Clients MUST/SHOULD RECOMMENDED be able to process the subtypes. Clients MUST/SHOULD RECOMMENDED be able to process the subtypes. Clients MUST/SHOULD RECOMMENDED be able to process the subtypes.

Use of a predefined subtypes is a prefixes in JSON Schema by `pattern`/`format`.

Process Graphs

An refined about a **process graph** is a chain of processes with nodes/n stacks for their parameters. Internally, a process graph is defined by a graph of connected processes, which exactly one node returns the result of the process.

`process_graph` in the example above, the process graph that is used to reference the process graph. Each process in the process graph has a unique identifier (`process_1`, `process_2`, `process_3`, `process_4`, `process_5`) within the process graph. Process nodes identifiers can also be assigned and can be referenced from child or parent process graphs. Circular references are not allowed.

Note: Also consider the Process Graph Syntax for more complex examples.

Processes (Process Nodes)

A single node in a process graph is a specific instance of a process is defined as follows:

```
{
  "process_id": "add",
  "arguments": {
    "x": 1,
    "y": 2
  }
}
```

A process node MUST always contain two properties named `process_id` and `arguments`. A MAY contains a `description`.

One of the nodes in a map of processes (the last one) MUST have the `result` flag set to `true`. Only one process node MUST/SHOULD RECOMMENDED be specified to be the result. Processes that are not connected to the result process MUST/SHOULD RECOMMENDED be ignored. Process node identifiers can only contain lowercase letters, numbers and underscores (`_`). Process node identifiers can also be prefixed with `from_` or `to_` to indicate that they are a process that is passed to or from a child process.

Arguments

A process can have an arbitrary number of arguments. These name and value are specified in the `arguments` property as an object of key/value pairs.

```
{
  "process_id": "filter_temporal",
  "arguments": {
    "data": {"from_node": "load_collection"},
    "extent": ["2018-01-01", "2018-02-01"]
  }
}
```

- The specified data types are the native data types supported by JSON, except for `/definitions/datacube`, `/definitions/process_graph`.
- Objects can be referenced to have keys with the following reserved names:
  - `from_node` for a child process node (`process_1`)
  - `from_parameter` used for referencing a parameter passed to the parent process.
  - `type` and `schema` for objects that `/definitions/parameters`.
- Optional parameters can be omitted if they are not required for the process specified by `process_id`.
- Parameters `/definitions/parameters` are specified in the `/processes` response.

Accessing results of other process nodes

A value of type `/definitions/process` is an object with keys `from_node` and `from_parameter`. `from_node` is a processing node.

```
{
  "process_id": "filter_temporal",
  "arguments": {
    "data": {"from_node": "load_collection"},
    "extent": ["2018-01-01", "2018-02-01"]
  }
}
```

Data cube is accessed from `from_parameter` and `from_node` is the node's name that contains a process node to be passed as argument. The `/definitions/parameters` `/definitions/process` is simply accepted and can also be referenced from child or parent process graphs. Circular references are not allowed.

Child processes

Processes can call other processes, which is similar to the concept that when programming functions can call functions. Child processes are specified in the `/definitions/process` property of the parent process.

Child processes can be specified in the parent process, which is a set of such a child process. They can be referenced in parent's `/definitions/process` property. Child processes are specified in the `/definitions/process` property of the parent process. Child processes are specified in the `/definitions/process` property of the parent process. Child processes are specified in the `/definitions/process` property of the parent process.

Process Parameters

A process can have an arbitrary number of parameters. These name and value are specified in the `parameters` property as an object of key/value pairs.

```
{
  "process_id": "filter_temporal",
  "parameters": [
    {"name": "data", "schema": {"type": "datacube"}},
    {"name": "extent", "schema": {"type": "array", "subtype": "temporal-interval"}}
  ]
}
```

- The specified data types are the native data types supported by JSON, except for `/definitions/datacube`, `/definitions/process_graph`.
- Objects can be referenced to have keys with the following reserved names:
  - `from_node` for a child process node (`process_1`)
  - `from_parameter` used for referencing a parameter passed to the parent process.
  - `type` and `schema` for objects that `/definitions/parameters`.
- Optional parameters can be omitted if they are not required for the process specified by `process_id`.
- Parameters `/definitions/parameters` are specified in the `/processes` response.

Accessing results of other process nodes

A value of type `/definitions/process` is an object with keys `from_node` and `from_parameter`. `from_node` is a processing node.

```
{
  "process_id": "filter_temporal",
  "arguments": {
    "data": {"from_node": "load_collection"},
    "extent": ["2018-01-01", "2018-02-01"]
  }
}
```

Data cube is accessed from `from_parameter` and `from_node` is the node's name that contains a process node to be passed as argument. The `/definitions/parameters` `/definitions/process` is simply accepted and can also be referenced from child or parent process graphs. Circular references are not allowed.

Child processes

Processes can call other processes, which is similar to the concept that when programming functions can call functions. Child processes are specified in the `/definitions/process` property of the parent process.

Child processes can be specified in the parent process, which is a set of such a child process. They can be referenced in parent's `/definitions/process` property. Child processes are specified in the `/definitions/process` property of the parent process. Child processes are specified in the `/definitions/process` property of the parent process. Child processes are specified in the `/definitions/process` property of the parent process.

Process Variables

A process can define variables, which are specified in the `variables` property of the process. Variables can be used in the process graph by referencing them with `{"from_variable": "variable_name"}`. Variables are scoped to the process graph and can be used in any nested child processes.

```
{
  "process_id": "example",
  "variables": [
    {"name": "threshold", "schema": {"type": "number"}, "default": 0.5},
    {"name": "band", "schema": {"type": "string"}, "default": "red"}
  ],
  "process_graph": {
    ...
    "filter": {"from_variable": "threshold"}
    ...
    "select": {"from_variable": "band"}
    ...
  }
}
```

- Variable names MUST be unique within the process.
- Variable names can only contain lowercase letters, numbers and underscores (`_`).
- Variables can be used in any process parameter that accepts a value matching the variable's `schema`.
- Variables can be used in nested child processes.
- Variables can be used in the `/processes` response.
- Variables can be used in the `/processes` response.
- Variables can be used in the `/processes` response.

Capabilities

Account Management

EO Data Discovery

Process Discovery

Data Retrieval/Processing

Job Control

Batch Jobs

Secondary Services

File Storage

```
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF xmlns="http://www.w3.org/2005/sparql-results#" />
```

Accessing process parameters

A SHACL shape that deals with a DMN decision can make it useful to access process parameters available in the DMN Spec. Processes in the DMN process model can access these parameters by using the `dmnx:processVariables` property. The following example shows how to access the parameters by using the `dmnx:processVariables` property.

```
PREFIX dmnx: <http://www.omg.org/spec/DMN/ns/interchange/>
PREFIX dmnt: <http://www.omg.org/spec/DMN/ns/interchange/typeRepresentation/>
PREFIX rdf: <http://www.w3.org/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX sh: <http://www.w3.org/ns/shacl#>
PREFIX ex: <http://example.com/ns#>

ex:ProcessVariableShape
    a sh:NodeShape ;
    sh:targetClass dmnx:ProcessVariables ;
    sh:property [
        sh:path dmnx:processVariables ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:node ex:ProcessVariableNodeShape ;
    ] .

ex:ProcessVariableNodeShape
    a sh:NodeShape ;
    sh:property [
        sh:path rdf:type ;
        sh:hasValue dmnt:Boolean ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] ;
    sh:property [
        sh:path rdfs:comment ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:datatype xsd:string ;
        sh:pattern "^This decision is processed.$"
        sh:flags "i"
    ] .
```

`dmnx:processVariables` specifies a single Process Variables property, which is a part of the specification. `dmnx:processVariables` is a DMN Extension Property.

Important: `dmnx:processVariables` is a DMN Extension Property. `dmnx:processVariables` is a DMN Extension Property. `dmnx:processVariables` can be any processes from the DMN Spec.

The rule for the assessment will be assessed as follows:

- It is ignored by the specific operation value is used. This means the assessment value is treated nothing but the current value and the assessment value is treated passed to a variable processes. Processes in the DMN process model can access these parameters by using the `dmnx:processVariables` property. It is not available for most properties. Most properties are available for most properties. Most properties are available for most properties. Most properties are

Information about the authenticated user

GET /v1

Lists information about the authenticated user such as the user id. The integration MAY return less data than specified by the scopes granted by the access token. The integration MAY return more data than specified by the scopes granted, but only if the resource owner has granted their consent and/or authorization. The integration MAY return data for other users than the authenticated user, but only if the resource owner has granted their consent and/or authorization. The integration MAY return data for other users than the authenticated user, but only if the access token is issued by the Notion workspace owner.

It is RECOMMENDED to implement the authentication method for non-public services only.

AUTHORIZATION `Bearer`

Responses

200 Information about the authenticated user successfully retrieved.

application/json

object (User)
Any of:

- object (string): Value capture_value. Constant. Always "user" (which is the value for the Bearer object in the underlying API).
- id (string): Unique identifier for this user.
- name (string): Optional. Sometimes set for people and bots.
- avatar_url (string): Optional. Sometimes set for people and bots.
- type (enum): Optional. Sometimes set for people and bots. Possible values: "person", "bot".
- person (object): Optional. Sometimes set for people.
  - email (string): Optional. Only returned if the integration has been granted the "Read user information including email addresses" capability.
- bot (object): Optional. Sometimes set for bots.
  - owner (object): Indicates the ownership of the bot. Possible values: "workspace", "user".
  - workspace_name (string): Optional. Only returned if the integration has been granted the "Read user information without email addresses" capability.

For additional response types see the Pets of previous versions page or contact us.

401 Unauthorized

application/json

object (Unauthorized)
Extends: object (Error)

- object (string): Value capture_value. Constant. Always "error".
- status (integer): Value capture_value. Constant. Always 401.
- code (enum): Value capture_value. Possible values: "unauthorized".
- message (string): A human-readable error message.
- additional_data (object): Optional. Machine-readable data that may be helpful in resolving the error.

For a response schema, see Pets of previous versions or contact us.

429 Rate limited.

application/json

object (RateLimited)
Extends: object (Error)

- object (string): Value capture_value. Constant. Always "error".
- status (integer): Value capture_value. Constant. Always 429.
- code (enum): Value capture_value. Possible values: "rate_limited".
- message (string): A human-readable error message.
- additional_data (object): Optional. Machine-readable data that may be helpful in resolving the error.

For a response schema, see Pets of previous versions or contact us.

500 Internal server error.

application/json

object (InternalServerError)
Extends: object (Error)

- object (string): Value capture_value. Constant. Always "error".
- status (integer): Value capture_value. Constant. Always 500.
- code (enum): Value capture_value. Possible values: "internal_server_error".
- message (string): A human-readable error message.
- additional_data (object): Optional. Machine-readable data that may be helpful in resolving the error.

For a response schema, see Pets of previous versions or contact us.

503 Service unavailable.

application/json

object (ServiceUnavailable)
Extends: object (Error)

- object (string): Value capture_value. Constant. Always "error".
- status (integer): Value capture_value. Constant. Always 503.
- code (enum): Value capture_value. Possible values: "service_unavailable".
- message (string): A human-readable error message.
- additional_data (object): Optional. Machine-readable data that may be helpful in resolving the error.

For a response schema, see Pets of previous versions or contact us.

Response samples

200 | 401 | 429

Copy Expand all Collapse all

```
{
  "object": "user",
  "id": "d40e767c-d7af-4b18-a86d-55c38f499dfb",
  "name": null,
  "avatar_url": null,
  "type": "bot",
  "bot": {
    "owner": {
      "type": "workspace",
      "workspace": true
    },
    "workspace_name": null
  }
}
```

EO Data Discovery

These endpoints allow to list the collections that are available at the hub and which can be used as data sources for data processing.

STAC

For data discovery of Earth Observation Collections at the hub-level, sentinel-hub-py complements with the specifications Spatial-Temporal Asset Catalog (STAC) and OGC API - Features. STAC is an API specification for implementing the data discovery endpoints of sentinel-hub-py produced with STAC Browser, QGIS, pystac-client, stac-fastapi, stac-server, stac-vue, and many other tools.

The data discovery endpoints GET /collections and GET /collections/{collectionId} are compatible with OGC API - Features and STAC. Both specifications define additional endpoints and parameters. However, STAC is more strict and has more mandatory requirements and thus can be used for API implementation. A large set of plugins for compatibility is available for a quick implementation of STAC on top of existing OGC API - Features implementations.

Important: STAC Item/Search and STAC API are different specifications and have different version numbers. The current version of STAC Item/Search is 1.0.0 and of STAC API - Features is 1.0.0. STAC API - Features is based on OGC API - Features - Part 1: Core 1.0 and implements a subset of OGC API - Features - Part 2: Coordinate reference systems by reference.

Content Extensions

STAC has several extensions that can be used to better describe your data. Clients and servers can use these extensions to indicate what they support. Extensions can be identified by their JSON-LD schemas.

Some commonly used extensions that are relevant for datasets exposed through the sentinel-hub-py API are:

- Electromagnetic Spectrum (eo)
- EO Products (eo-products)
- Processing (processing)
- Projection (proj)
- SAR (sar)
- Sensor Model (sensor-model)
- Statistics (statistics)
- Scientific Citation (scientific)

Provider data for download

A good link to provide your data for download is additive to offering the chosen processing service, you provide your data for download via STAC. Clients can use the provided links.

Recommended extensions for providing data for download are Cloud-Optimized Geospatial Formats Extension to benefit from the STAC ecosystem and allow searching for items you can also implement Stac-fastapi-geoparquet or Stac-fastapi-elasticsearch-dev to benefit from fast queries.

API Extensions

STAC API has several extensions that can be implemented on top of the core/OGC API to extend the API functionality, e.g. for searching.

Basic metadata for all datasets

GET /v1/collections

Lists available collections with basic information.

It is strongly RECOMMENDED to follow the pagination and only working page optional values from the objects in `links` (`next`, `prev`, `first`, `last`) and `pagination` properties. To get the `pagination` properties, the client MUST set the `pagination` parameter to `true`.

The collections in compliance with OGC API Common part 1: Core and OGC API - Features - Part 1: Core, OGC Implementing STAC extension are implemented in addition to what is documented here.

Note: Although it is possible to query data from all collections, it is not recommended to do so. Instead, the client should first filter the collections that are available to store which they provide collections.

AUTHORIZATION `Bearer`

Responses

200 Basic information for all datasets.

application/json

object (Collections)

- links* (array of objects (LinkExternal)): Links to other resources. For example links for pagination or links relevant to the current resource (resource URI). Values: Any string MAY NOT be empty.
  - Expand/collapse to implement the following: (array): Unknown type apply.
  - rel* (enum): REQUIRED. A link to the next page, or last in the last page. Values: "next", "prev", "first", "last", "self", "alternate", "collection", "parent", "root".
  - uri* (string): REQUIRED. A URI-Reference. Values: Any string MAY NOT be empty.
  - type (enum): OPTIONAL. A link to the next page, except or the last page. Values: "application/geo+json", "application/ld+json", "application/vnd.oai.openapi+json", "application/vnd.oai.openapi+json;version=3.0", "application/vnd.oai.openapi+json;version=3.1", "application/vnd.oai.openapi;version=3.0", "application/vnd.oai.openapi;version=3.1", "application/schema+json", "application/vnd.oai.openapi", "application/openapi+json", "application/vnd.oai.openapi;version=2.0", "application/openapi+json;version=2.0", "application/vnd.oai.openapi+ndjson", "application/vnd.oai.openapi+ndjson;version=3.0", "application/vnd.oai.openapi+ndjson;version=3.1", "application/vnd.oai.openapi+ndjson;version=2.0", "application/vnd.oai.openapi+ndjson;version=2.0", "application/ndjson", "application/ndjson;version=2.0", "application/ndjson;version=3.0", "application/ndjson;version=3.1", "application/ndjson;version=2.0", "application/ndjson;version=2.0", "application/ndjson;version=2.0

List all user-defined processes

GET /process_graphs

List all user-defined process graphs of the authenticated user that are stored at the backend.

It is **strongly RECOMMENDED** to keep the listing large values minimal and to store the objects in `processes` (e.g. the `summary`, `description` and `links` properties). To get the full metadata for a user-defined process clients MUST request `GET /process_graphs/{process_graph_id}`.

AUTHORIZATION REQUIRED: Basic or Bearer

QUERY PARAMETERS

- limit *integer* — A limit on the number of results (or page size). Excluded from top level collection objects.

  Default: `10`; Minimum: `1`

  Pagination is OPTIONAL. Each service or client may set a default for this parameter if omitted. The value must preferably be a reasonable number for interactive requests and clients MUST support values up to 10000.

  If the requested is supported, the `links` array MUST be used to navigate to the next page. If the backend chooses to return less than the requested `limit`, Open Data Clients cannot be supported — `links` MUST be used to navigate to the next page. Clients may either stop paginating or adjust the `limit` parameter in the next request.

  If the requested `limit` is not supported, the backend MUST return up to `n` collections.

- offset *string* — Skip the first `n` elements from the collection.

  Pagination is implemented like following: `GET /collections?offset=100&limit=50`

  - `Link: <?offset=150&limit=50>; rel="next"` — Skip the next 100 elements and return the next 50 elements.
  - `Link: <?offset=0&limit=50>; rel="prev"` — Skip the first 50 elements and return the first 50 elements.

  For additional relation types see the IANA list of process relation types in openeo/processes.

  Default: `0`; Minimum: `0`

Responses

- 200 A list of user-defined processes stored at the backend.

  application/json

  - processes *array of objects (Process Summary Metadata)* — A list of user-defined processes.

    Array of objects (Data validation):

    - id *string* — The process identifier.

      The process identifier for a user-defined process is optional and MUST be unique for the requested namespace.

      Implementations are advised to use a random UUIDs, random strings or hashes for this purpose.

      Pattern: `^[A-Za-z0-9_-]+$`; Example: `"myprocess"`
    - summary *string* — A short summary of the process.

      Example: `"Calculate NDVI"`
    - description *string* — Detailed description to explain the entity.

      Commonmark 0.30 is allowed but not recommended.

      Example: `"Computes the Normalized Difference Vegetation Index (NDVI) based on the provided bands."`
    - categories *array of unique items (string)* — A list of categories.

      Example: `["cubes", "vector-cubes"]`
    - parameters *array of objects (Process Parameter)* — The parameters that can be passed to the process.

      Array of objects (Data validation):

      - name *string* — The name of the parameter.

        The name of the parameter is unique for the process.

        Pattern: `^[A-Za-z0-9_-]+$`; Example: `"data"`
      - description *string* — Detailed description to explain the entity.

        Commonmark 0.30 is allowed but not recommended.

        Example: `"Specifies the underlying raster data cube."`
      - schema *object or array of objects (Process Schema)* — JSON Schema(s) describing the parameter value.

        Example: `{"type": "number", "minimum": 0}`
      - optional *boolean* — Default: `false`

        Indicates that the parameter is optional.
      - deprecated *boolean* — Default: `false`

        Indicates that the parameter is deprecated with the potential to be removed in the future. Clients should avoid using this parameter.
      - experimental *boolean* — Default: `false`

        Indicates that the parameter is experimental. Clients should be aware that the parameter may change or disappear without a deprecation period.
      - default *any* — Default value for the parameter.

        The process may define a default value for the parameter.
      - delete_if_null *boolean* — Default: `false`

        If set to `true`, the parameter can be set to `null` to remove the parameter from the process graph.
    - parameter_order *array of unique items (string)* — Defines the order in which parameters are expected when calling the process.

      If omitted, all required parameters are expected to be specified in alphabetical order.
    - returns *object or array of objects (Process Schema)* — Describes the data to be returned by the process.

      Example: `{"type": "object", "subtype": "raster-cube"}`
    - exceptions *object* — Errors that may occur when calling the process.

      Example: `{"TooSmall": {"message": "Value must be at least 1"}}`
    - examples *array of objects (Example)* — Examples, may be used for unit tests.

      Array of objects (Data validation):

      - title *string* — A short summary of the process.

        Example: `"Calculate NDVI"`
      - description *string* — Detailed description to explain the entity.

        Commonmark 0.30 is allowed but not recommended.

        Example: `"Computes the Normalized Difference Vegetation Index (NDVI) based on the provided bands."`
      - arguments *object* — Specifies the process parameters and their values.

        Example: `{"data": {"from_node": "loadcollection1"}, "bands": {"from_parameter": "data"}, "nir": "nir", "red": "red"}`
      - returns *any* — Describes the expected return value.

        Example: `{"type": "object", "subtype": "raster-cube"}`
    - links *array of objects (Link)* — Links to related resources.

      Array of objects (Data validation):

      - href *string* — Target URI.

        Example: `"https://example.com/process"`
      - rel *string* — Relation type.

        Example: `"parent"`
      - type *string* — Media type hint.

        Example: `"application/geo+json"`
      - title *string* — Optional link text.

        Example: `"Parent collection"`
    - parameter_order *array of unique items (string)* — Defines the order in which parameters are expected when calling the process.

      If omitted, all required parameters are expected to be specified in alphabetical order.
    - returns *object or array of objects (Process Schema)* — Describes the data to be returned by the process.

      Example: `{"type": "object", "subtype": "raster-cube"}`
    - exceptions *object* — Errors that may occur when calling the process.

      Example: `{"TooSmall": {"message": "Value must be at least 1"}}`
    - examples *array of objects (Example)* — Examples, may be used for unit tests.

      Array of objects (Data validation):

      - title *string* — A short summary of the process.

        Example: `"Calculate NDVI"`
      - description *string* — Detailed description to explain the entity.

        Commonmark 0.30 is allowed but not recommended.

        Example: `"Computes the Normalized Difference Vegetation Index (NDVI) based on the provided bands."`
      - arguments *object* — Specifies the process parameters and their values.

        Example: `{"data": {"from_node": "loadcollection1"}, "bands": {"from_parameter": "data"}, "nir": "nir", "red": "red"}`
      - returns *any* — Describes the expected return value.

        Example: `{"type": "object", "subtype": "raster-cube"}`
  - links *array of objects (Link)* — Links to related resources.

    Array of objects (Data validation):

    - href *string* — Target URI.

      Example: `"https://example.com/process"`
    - rel *string* — Relation type.

      Example: `"parent"`
    - type *string* — Media type hint.

      Example: `"application/geo+json"`
    - title *string* — Optional link text.

      Example: `"Parent collection"`

Response samples

200 — application/json

```
{
  "processes": [
    {
      "id": "myprocess",
      "summary": "Calculate NDVI",
      "description": "Computes the Normalized Difference Vegetation Index (NDVI) based on the provided bands.",
      "categories": [
        "cubes",
        "vector-cubes"
      ],
      "parameters": [
        {
          "name": "data",
          "description": "Specifies the underlying raster data cube.",
          "schema": {
            "type": "number",
            "minimum": 0
          },
          "optional": false,
          "deprecated": false,
          "experimental": false,
          "default": null,
          "delete_if_null": false
        }
      ],
      "parameter_order": [
        "data"
      ],
      "returns": {
        "type": "object",
        "subtype": "raster-cube"
      },
      "exceptions": {
        "TooSmall": {
          "message": "Value must be at least 1"
        }
      },
      "examples": [
        {
          "title": "Calculate NDVI",
          "description": "Computes the Normalized Difference Vegetation Index (NDVI) based on the provided bands.",
          "arguments": {
            "data": {
              "from_node": "loadcollection1"
            },
            "bands": {
              "from_parameter": "data"
            },
            "nir": "nir",
            "red": "red"
          },
          "returns": {
            "type": "object",
            "subtype": "raster-cube"
          }
        }
      ],
      "parameter_order": [
        "data"
      ],
      "returns": {
        "type": "object",
        "subtype": "raster-cube"
      },
      "exceptions": {
        "TooSmall": {
          "message": "Value must be at least 1"
        }
      },
      "examples": [
        {
          "title": "Calculate NDVI",
          "description": "Computes the Normalized Difference Vegetation Index (NDVI) based on the provided bands.",
          "arguments": {
            "data": {
              "from_node": "loadcollection1"
            },
            "bands": {
              "from_parameter": "data"
            },
            "nir": "nir",
            "red": "red"
          },
          "returns": {
            "type": "object",
            "subtype": "raster-cube"
          }
        }
      ],
      "links": [
        {
          "href": "https://example.com/process",
          "rel": "parent",
          "type": "application/geo+json",
          "title": "Parent collection"
        }
      ]
    }
  ],
  "links": [
    {
      "href": "https://example.com/process",
      "rel": "parent",
      "type": "application/geo+json",
      "title": "Parent collection"
    }
  ]
}
```

Full metadata for a user-defined process

GET /process_graphs/{process_graph_id}

Lists all information about a user-defined process, including its process graph.

AUTHORIZATION REQUIRED: Basic or Bearer

PATH PARAMETERS

- process_graph_id *string (Identifier)* *required* — User-defined process identifier.

  Pattern: `^[A-Za-z0-9_-]+$`; Example: `"myprocess"`

  The user-unique identifier for a user-defined process.

Responses

- 200 Full data of a user-defined process.

  application/json

  - id *string* — A short summary of the process.

    Example: `"myprocess"`
  - summary *string* — A short summary of the process.

    Example: `"Calculate NDVI"`
  - description *string* — Detailed description to explain the entity.

    Commonmark 0.30 is allowed but not recommended.

    Example: `"Computes the Normalized Difference Vegetation Index (NDVI) based on the provided bands."`
  - categories *array of unique items (string)* — A list of categories.

    Example: `["cubes", "vector-cubes"]`
  - parameters *array of objects (Process Parameter)* — The parameters that can be passed to the process.

    Array of objects (Data validation):

    - name *string* — The name of the parameter.

      The name of the parameter is unique for the process.

      Pattern: `^[A-Za-z0-9_-]+$`; Example: `"data"`
    - description *string* — Detailed description to explain the entity.

      Commonmark 0.30 is allowed but not recommended.

      Example: `"Specifies the underlying raster data cube."`
    - schema *object or array of objects (Process Schema)* — JSON Schema(s) describing the parameter value.

      Example: `{"type": "number", "minimum": 0}`
    - optional *boolean* — Default: `false`

      Indicates that the parameter is optional.
    - deprecated *boolean* — Default: `false`

      Indicates that the parameter is deprecated with the potential to be removed in the future. Clients should avoid using this parameter.
    - experimental *boolean* — Default: `false`

      Indicates that the parameter is experimental. Clients should be aware that the parameter may change or disappear without a deprecation period.
    - default *any* — Default value for the parameter.

      The process may define a default value for the parameter.
    - delete_if_null *boolean* — Default: `false`

      If set to `true`, the parameter can be set to `null` to remove the parameter from the process graph.
  - parameter_order *array of unique items (string)* — Defines the order in which parameters are expected when calling the process.

    If omitted, all required parameters are expected to be specified in alphabetical order.
  - returns *object or array of objects (Process Schema)* — Describes the data to be returned by the process.

    Example: `{"type": "object", "subtype": "raster-cube"}`
  - exceptions *object* — Errors that may occur when calling the process.

    Example: `{"TooSmall": {"message": "Value must be at least 1"}}`
  - examples *array of objects (Example)* — Examples, may be used for unit tests.

    Array of objects (Data validation):

    - title *string* — A short summary of the process.

      Example: `"Calculate NDVI"`
    - description *string* — Detailed description to explain the entity.

      Commonmark 0.30 is allowed but not recommended.

      Example: `"Computes the Normalized Difference Vegetation Index (NDVI) based on the provided bands."`
    - arguments *object* — Specifies the process parameters and their values.

      Example: `{"data": {"from_node": "loadcollection1"}, "bands": {"from_parameter": "data"}, "nir": "nir", "red": "red"}`
    - returns *any* — Describes the expected return value.

      Example: `{"type": "object", "subtype": "raster-cube"}`
  - links *array of objects (Link)* — Links to related resources.

    Array of objects (Data validation):

    - href *string* — Target URI.

      Example: `"https://example.com/process"`
    - rel *string* — Relation type.

      Example: `"parent"`
    - type *string* — Media type hint.

      Example: `"application/geo+json"`
    - title *string* — Optional link text.

      Example: `"Parent collection"`

Response samples

200 — application/json

```
{
  "id": "myprocess",
  "summary": "Calculate NDVI",
  "description": "Computes the Normalized Difference Vegetation Index (NDVI) based on the provided bands.",
  "categories": [
    "cubes",
    "vector-cubes"
  ],
  "parameters": [
    {
      "name": "data",
      "description": "Specifies the underlying raster data cube.",
      "schema": {
        "type": "number",
        "minimum": 0
      },
      "optional": false,
      "deprecated": false,
      "experimental": false,
      "default": null,
      "delete_if_null": false
    }
  ],
  "parameter_order": [
    "data"
  ],
  "returns": {
    "type": "object",
    "subtype": "raster-cube"
  },
  "exceptions": {
    "TooSmall": {
      "message": "Value must be at least 1"
    }
  },
  "examples": [
    {
      "title": "Calculate NDVI",
      "description": "Computes the Normalized Difference Vegetation Index (NDVI) based on the provided bands.",
      "arguments": {
        "data": {
          "from_node": "loadcollection1"
        },
        "bands": {
          "from_parameter": "data"
        },
        "nir": "nir",
        "red": "red"
      },
      "returns": {
        "type": "object",
        "subtype": "raster-cube"
      }
    }
  ],
  "parameter_order": [
    "data"
  ],
  "returns": {
    "type": "object",
    "subtype": "raster-cube"
  },
  "exceptions": {
    "TooSmall": {
      "message": "Value must be at least 1"
    }
  },
  "examples": [
    {
      "title": "Calculate NDVI",
      "description": "Computes the Normalized Difference Vegetation Index (NDVI) based on the provided bands.",
      "arguments": {
        "data": {
          "from_node": "loadcollection1"
        },
        "bands": {
          "from_parameter": "data"
        },
        "nir": "nir",
        "red": "red"
      },
      "returns": {
        "type": "object",
        "subtype": "raster-cube"
      }
    }
  ],
  "links": [
    {
      "href": "https://example.com/process",
      "rel": "parent",
      "type": "application/geo+json",
      "title": "Parent collection"
    }
  ]
}
```

Store a user-defined process

PUT /process_graphs/{process_graph_id}

Stores a user-defined process with a process graph that can be reused in other processes.

If a process with the specified `process_graph_id` exists, the process is fully replaced. For an existing process, partial updates are allowed by sending only the properties to be changed.

Partially updating a user-defined process is OPTIONAL and may not be supported by all backends.

To simplify re-defining user-defined processes, the property `id` can be part of the request body, if it is specified in the URL path and matches with the value from the URL.

AUTHORIZATION REQUIRED: Basic or Bearer

PATH PARAMETERS

- process_graph_id *string (Identifier)* *required* — User-defined process identifier.

  Pattern: `^[A-Za-z0-9_-]+$`; Example: `"myprocess"`

  The user-unique identifier for a user-defined process.

BODY PARAMETERS (application/json)

- id *string* — A short summary of the process.

  Example: `"myprocess"`
- summary *string* — A short summary of the process.

  Example: `"Calculate NDVI"`
- description *string* — Detailed description to explain the entity.

  Commonmark 0.30 is allowed but not recommended.

  Example: `"Computes the Normalized Difference Vegetation Index (NDVI) based on the provided bands."`
- categories *array of unique items (string)* — A list of categories.

  Example: `["cubes", "vector-cubes"]`
- parameters *array of objects (Process Parameter)* — The parameters that can be passed to the process.

  Array of objects (Data validation):

  - name *string* — The name of the parameter.

    The name of the parameter is unique for the process.

    Pattern: `^[A-Za-z0-9_-]+$`; Example: `"data"`
  - description *string* — Detailed description to explain the entity.

    Commonmark 0.30 is allowed but not recommended.

    Example: `"Specifies the underlying raster data cube."`
  - schema *object or array of objects (Process Schema)* — JSON Schema(s) describing the parameter value.

    Example: `{"type": "number", "minimum": 0}`
  - optional *boolean* — Default: `false`

    Indicates that the parameter is optional.
  - deprecated *boolean* — Default: `false`

    Indicates that the parameter is deprecated with the potential to be removed in the future. Clients should avoid using this parameter.
  - experimental *boolean* — Default: `false`

    Indicates that the parameter is experimental. Clients should be aware that the parameter may change or disappear without a deprecation period.
  - default *any* — Default value for the parameter.

    The process may define a default value for the parameter.
  - delete_if_null *boolean* — Default: `false`

    If set to `true`, the parameter can be set to `null` to remove the parameter from the process graph.
- parameter_order *array of unique items (string)* — Defines the order in which parameters are expected when calling the process.

  If omitted, all required parameters are expected to be specified in alphabetical order.
- returns *object or array of objects (Process Schema)* — Describes the data to be returned by the process.

  Example: `{"type": "object", "subtype": "raster-cube"}`
- exceptions *object* — Errors that may occur when calling the process.

  Example: `{"TooSmall": {"message": "Value must be at least 1"}}`
- examples *array of objects (Example)* — Examples, may be used for unit tests.

  Array of objects (Data validation):

  - title *string* — A short summary of the process.

    Example: `"Calculate NDVI"`
  - description *string* — Detailed description to explain the entity.

    Commonmark 0.30 is allowed but not recommended.

    Example: `"Computes the Normalized Difference Vegetation Index (NDVI) based on the provided bands."`
  - arguments *object* — Specifies the process parameters and their values.

    Example: `{"data": {"from_node": "loadcollection1"}, "bands": {"from_parameter": "data"}, "nir": "nir", "red": "red"}`
  - returns *any* — Describes the expected return value.

    Example: `{"type": "object", "subtype": "raster-cube"}`
- links *array of objects (Link)* — Links to related resources.

  Array of objects (Data validation):

  - href *string* — Target URI.

    Example: `"https://example.com/process"`
  - rel *string* — Relation type.

    Example: `"parent"`
  - type *string* — Media type hint.

    Example: `"application/geo+json"`
  - title *string* — Optional link text.

    Example: `"Parent collection"`

Response samples

Request — application/json

```
{
  "id": "myprocess",
  "summary": "Calculate NDVI",
  "description": "Computes the Normalized Difference Vegetation Index (NDVI) based on the provided bands.",
  "categories": [
    "cubes",
    "vector-cubes"
  ],
  "parameters": [
    {
      "name": "data",
      "description": "Specifies the underlying raster data cube.",
      "schema": {
        "type": "number",
        "minimum": 0
      },
      "optional": false,
      "deprecated": false,
      "experimental": false,
      "default": null,
      "delete_if_null": false
    }
  ],
  "parameter_order": [
    "data"
  ],
  "returns": {
    "type": "object",
    "subtype": "raster-cube"
  },
  "exceptions": {
    "TooSmall": {
      "message": "Value must be at least 1"
    }
  },
  "examples": [
    {
      "title": "Calculate NDVI",
      "description": "Computes the Normalized Difference Vegetation Index (NDVI) based on the provided bands.",
      "arguments": {
        "data": {
          "from_node": "loadcollection1"
        },
        "bands": {
          "from_parameter": "data"
        },
        "nir": "nir",
        "red": "red"
      },
      "returns": {
        "type": "object",
        "subtype": "raster-cube"
      }
    }
  ],
  "parameter_order": [
    "data"
  ],
  "returns": {
    "type": "object",
    "subtype": "raster-cube"
  },
  "exceptions": {
    "TooSmall": {
      "message": "Value must be at least 1"
    }
  },
  "examples": [
    {
      "title": "Calculate NDVI",
      "description": "Computes the Normalized Difference Vegetation Index (NDVI) based on the provided bands.",
      "arguments": {
        "data": {
          "from_node": "loadcollection1"
        },
        "bands": {
          "from_parameter": "data"
        },
        "nir": "nir",
        "red": "red"
      },
      "returns": {
        "type": "object",
        "subtype": "raster-cube"
      }
    }
  ],
  "links": [
    {
      "href": "https://example.com/process",
      "rel": "parent",
      "type": "application/geo+json",
      "title": "Parent collection"
    }
  ]
}
```

Response samples

200 — application/json

```
{
  "id": "myprocess",
  "summary": "Calculate NDVI",
  "description": "Computes the Normalized Difference Vegetation Index (NDVI) based on the provided bands.",
  "categories": [
    "cubes",
    "vector-cubes"
  ],
  "parameters": [
    {
      "name": "data",
      "description": "Specifies the underlying raster data cube.",
      "schema": {
        "type": "number",
        "minimum": 0
      },
      "optional": false,
      "deprecated": false,
      "experimental": false,
      "default": null,
      "delete_if_null": false
    }
  ],
  "parameter_order": [
    "data"
  ],
  "returns": {
    "type": "object",
    "subtype": "raster-cube"
  },
  "exceptions": {
    "TooSmall": {
      "message": "Value must be at least 1"
    }
  },
  "examples": [
    {
      "title": "Calculate NDVI",
      "description": "Computes the Normalized Difference Vegetation Index (NDVI) based on the provided bands.",
      "arguments": {
        "data": {
          "from_node": "loadcollection1"
        },
        "bands": {
          "from_parameter": "data"
        },
        "nir": "nir",
        "red": "red"
      },
      "returns": {
        "type": "object",
        "subtype": "raster-cube"
      }
    }
  ],
  "parameter_order": [
    "data"
  ],
  "returns": {
    "type": "object",
    "subtype": "raster-cube"
  },
  "exceptions": {
    "TooSmall": {
      "message": "Value must be at least 1"
    }
  },
  "examples": [
    {
      "title": "Calculate NDVI",
      "description": "Computes the Normalized Difference Vegetation Index (NDVI) based on the provided bands.",
      "arguments": {
        "data": {
          "from_node": "loadcollection1"
        },
        "bands": {
          "from_parameter": "data"
        },
        "nir": "nir",
        "red": "red"
      },
      "returns": {
        "type": "object",
        "subtype": "raster-cube"
      }
    }
  ],
  "links": [
    {
      "href": "https://example.com/process",
      "rel": "parent",
      "type": "application/geo+json",
      "title": "Parent collection"
    }
  ]
}
```

Validate a user defined process (graph)

POST /v1/authorization/validate

Validates a user defined process written according to a user defined process in a convenient JSON format. (JSON Graph Format) and returns a machine readable version of the process.

Unlike all other the process graph is automatically current and the processes are supported by the current version of the authorization service. Unlike other APIs, this API does not require a specific version of authorization service to be specified. For example, a non-existing process may get rejected only by the back-end. The validation MUST NOT throw an error for unsupported processes.

Authorization service MAY reject a process if it violates the corresponding resource's JSONSchema.

Errors that prevent other during processing MAY NOT get reported e.g. if a referenced file is missing, or a remote resource is unavailable.

Authorization service MAY also provide an extra or stricter validation mode that rejects the process.

Please note that a validation always returns with HTTP status code 200. Error codes in the form of HTTP status codes are reserved for errors that happen before the validation logic is executed (e.g. if the user defined process is not recognized or unauthorized).

AUTHORIZATION: Bearer <token>

REQUEST BODY SCHEMA: application/json

Specifies the user defined process to be validated.

- `id` string, nullable
  Unique identifier of the process. Optional.
- `summary` string, nullable
  A short summary of what the process does.
- `description` string, nullable
  Detailed description to explain the workflow.
- `processDefinition` enum (required)
  Specifies the process graph to be validated. Currently, the following values are accepted: `JSON-Graph-Process-Definition` (JSON Graph Format) and `Cadence-Flow` (Cadence Flow). Other values may be accepted in other implementations.
  Enum: `"JSON-Graph-Process-Definition"` `"Cadence-Flow"`
- `process` object (required)
  A JSON representation of a process graph.
  - `nodes` array of objects (required)
    A list of nodes.
    - `id` string (required)
      Unique identifier for this node.
    - `type` enum (required)
      Specifies the type of node. Currently, the following values are accepted: `activity` (Activity), `decision` (Decision), `parallel` (Parallel), `loop` (Loop), `start` (Start), `end` (End), `fail` (Fail). Other values may be accepted in other implementations.
      Enum: `"activity"` `"decision"` `"parallel"` `"loop"` `"start"` `"end"` `"fail"`
    - `parameters` object, nullable
      Defines the parameters which are passed to this node. Optional.
      - `activityId` string
        Specifies the activity to be executed.
      - `activityName` string
        Human readable name of the activity. Optional.
      - `timeout` integer, nullable
        Specifies the maximum time in seconds the activity may take to complete before it times out. Optional.
      - `retryPolicy` object, nullable
        Defines how the activity should be retried if it fails. Optional.
        - `initialInterval` integer
          Specifies the initial wait time in seconds before the first retry attempt.
        - `backoffCoefficient` number
          Specifies the multiplier used to calculate the wait time between retry attempts.
        - `maximumInterval` integer, nullable
          Specifies the maximum wait time in seconds between retry attempts. Optional.
        - `maximumAttempts` integer
          Specifies the maximum number of attempts. Zero means unlimited.
    - `metadata` object, nullable
      Optional metadata for UI rendering. Ignored by the workflow engine.
    - `additionalProperties` object
    - `additionalProperties` object
  - `edges` array of objects (required)
    A list of edges.
    - `id` string (required)
      Unique identifier for this edge.
    - `source` string (required)
      Specifies the source node id.
    - `sourceHandle` enum, nullable
      Specifies the source handle. Optional. Currently, the following values are accepted: `success` (Success), `failure` (Failure), `loop` (Loop). Other values may be accepted in other implementations.
      Enum: `"success"` `"failure"` `"loop"`
    - `target` string (required)
      Specifies the target node id.
    - `metadata` object, nullable
      Optional metadata for UI rendering. Ignored by the workflow engine.
    - `additionalProperties` object
    - `additionalProperties` object
  - `additionalProperties` object
  - `additionalProperties` object
- `additionalProperties` object
- `additionalProperties` object

Response samples

200 Validation succeeded

Content-type: application/json

```
{
  "id": "",
  "summary": "",
  "description": "",
  "processDefinition": "JSON-Graph-Process-Definition",
  "process": {
    "nodes": [
      {
        "id": "",
        "type": "activity",
        "parameters": {
          "activityId": "",
          "activityName": "",
          "timeout": 0,
          "retryPolicy": {
            "initialInterval": 0,
            "backoffCoefficient": 0,
            "maximumInterval": 0,
            "maximumAttempts": 0
          }
        },
        "metadata": {
          "additionalProperties": {}
        }
      }
    ],
    "edges": [
      {
        "id": "",
        "source": "",
        "sourceHandle": "success",
        "target": "",
        "metadata": {
          "additionalProperties": {}
        }
      }
    ],
    "additionalProperties": {}
  }
}
```

Copy Expand all Collapse all

400 Bad request

Content-type: application/problem+json

```
{
  "type": "",
  "title": "",
  "status": 0,
  "detail": "",
  "instance": "",
  "errors": [
    {
      "pointer": "",
      "detail": ""
    }
  ]
}
```

Copy Expand all Collapse all

401 Unauthorized

Content-type: application/problem+json

```
{
  "type": "",
  "title": "",
  "status": 0,
  "detail": "",
  "instance": "",
  "errors": [
    {
      "pointer": "",
      "detail": ""
    }
  ]
}
```

Copy Expand all Collapse all

403 Forbidden

Content-type: application/problem+json

```
{
  "type": "",
  "title": "",
  "status": 0,
  "detail": "",
  "instance": "",
  "errors": [
    {
      "pointer": "",
      "detail": ""
    }
  ]
}
```

Copy Expand all Collapse all

500 Internal server error

Content-type: application/problem+json

```
{
  "type": "",
  "title": "",
  "status": 0,
  "detail": "",
  "instance": "",
  "errors": [
    {
      "pointer": "",
      "detail": ""
    }
  ]
}
```

Copy Expand all Collapse all

Process and download data synchronously

POST /v1/authorization/process

Validates a user defined process (JSON Graph Format) and executes it synchronously. The process is forced specified in the process graph. This endpoint can be used to generate small preview or test data. Authorization service MAY reject processes that are too complex or too long to be executed synchronously. Authorization service MAY reject the request if the process is not supported by the current version of authorization service. Authorization service MAY reject processes that violate the corresponding resource's JSONSchema. Authorization service MAY reject processes that are too complex or too long to be executed synchronously. Authorization service MAY reject the request if the process is not supported by the current version of authorization service.

AUTHORIZATION: Bearer <token>

REQUEST BODY SCHEMA: application/json

Specifies the job details, e.g. the user defined process and timing details.

- `id` string, nullable
  A short identifier for easing debugging and logs.
- `summary` string, nullable
  A short summary of what the process does. Optional.
- `description` string, nullable
  Detailed description to explain the workflow.
- `processDefinition` enum (required)
  Specifies the job details, e.g. the user defined process and timing details. Currently, the following values are accepted: `JSON-Graph-Process-Definition` (JSON Graph Format) and `Cadence-Flow` (Cadence Flow). Other values may be accepted in other implementations.
  Enum: `"JSON-Graph-Process-Definition"` `"Cadence-Flow"`
- `process` object (required)
  A JSON representation of a process graph.
  - `nodes` array of objects (required)
    A list of nodes.
    - `id` string (required)
      Unique identifier for this node.
    - `type` enum (required)
      Specifies the type of node. Currently, the following values are accepted: `activity` (Activity), `decision` (Decision), `parallel` (Parallel), `loop` (Loop), `start` (Start), `end` (End), `fail` (Fail). Other values may be accepted in other implementations.
      Enum: `"activity"` `"decision"` `"parallel"` `"loop"` `"start"` `"end"` `"fail"`
    - `parameters` object, nullable
      Defines the parameters which are passed to this node. Optional.
      - `activityId` string
        Specifies the activity to be executed.
      - `activityName` string
        Human readable name of the activity. Optional.
      - `timeout` integer, nullable
        Specifies the maximum time in seconds the activity may take to complete before it times out. Optional.
      - `retryPolicy` object, nullable
        Defines how the activity should be retried if it fails. Optional.
        - `initialInterval` integer
          Specifies the initial wait time in seconds before the first retry attempt.
        - `backoffCoefficient` number
          Specifies the multiplier used to calculate the wait time between retry attempts.
        - `maximumInterval` integer, nullable
          Specifies the maximum wait time in seconds between retry attempts. Optional.
        - `maximumAttempts` integer
          Specifies the maximum number of attempts. Zero means unlimited.
    - `metadata` object, nullable
      Optional metadata for UI rendering. Ignored by the workflow engine.
    - `additionalProperties` object
    - `additionalProperties` object
  - `edges` array of objects (required)
    A list of edges.
    - `id` string (required)
      Unique identifier for this edge.
    - `source` string (required)
      Specifies the source node id.
    - `sourceHandle` enum, nullable
      Specifies the source handle. Optional. Currently, the following values are accepted: `success` (Success), `failure` (Failure), `loop` (Loop). Other values may be accepted in other implementations.
      Enum: `"success"` `"failure"` `"loop"`
    - `target` string (required)
      Specifies the target node id.
    - `metadata` object, nullable
      Optional metadata for UI rendering. Ignored by the workflow engine.
    - `additionalProperties` object
    - `additionalProperties` object
  - `additionalProperties` object
  - `additionalProperties` object
- `additionalProperties` object
- `additionalProperties` object

Response samples

200 Process executed successfully

Content-type: application/octet-stream

```
{
  "id": "",
  "summary": "",
  "description": "",
  "processDefinition": "JSON-Graph-Process-Definition",
  "process": {
    "nodes": [
      {
        "id": "",
        "type": "activity",
        "parameters": {
          "activityId": "",
          "activityName": "",
          "timeout": 0,
          "retryPolicy": {
            "initialInterval": 0,
            "backoffCoefficient": 0,
            "maximumInterval": 0,
            "maximumAttempts": 0
          }
        },
        "metadata": {
          "additionalProperties": {}
        }
      }
    ],
    "edges": [
      {
        "id": "",
        "source": "",
        "sourceHandle": "success",
        "target": "",
        "metadata": {
          "additionalProperties": {}
        }
      }
    ],
    "additionalProperties": {}
  }
}
```

Copy Expand all Collapse all

400 Bad request

Content-type: application/problem+json

```
{
  "type": "",
  "title": "",
  "status": 0,
  "detail": "",
  "instance": "",
  "errors": [
    {
      "pointer": "",
      "detail": ""
    }
  ]
}
```

Copy Expand all Collapse all

401 Unauthorized

Content-type: application/problem+json

```
{
  "type": "",
  "title": "",
  "status": 0,
  "detail": "",
  "instance": "",
  "errors": [
    {
      "pointer": "",
      "detail": ""
    }
  ]
}
```

Copy Expand all Collapse all

403 Forbidden

Content-type: application/problem+json

```
{
  "type": "",
  "title": "",
  "status": 0,
  "detail": "",
  "instance": "",
  "errors": [
    {
      "pointer": "",
      "detail": ""
    }
  ]
}
```

Copy Expand all Collapse all

500 Internal server error

Content-type: application/problem+json

```
{
  "type": "",
  "title": "",
  "status": 0,
  "detail": "",
  "instance": "",
  "errors": [
    {
      "pointer": "",
      "detail": ""
    }
  ]
}
```

Copy Expand all Collapse all

List all batch jobs

GET /v1/authorization/batch

Lists all jobs submitted by current tenant.

It's strongly **RECOMMENDED** to keep the response size small by setting all optional size limits properly. Authorization service MAY ignore any of the limits. Authorization service MAY reject requests that are too complex and MAY support `?size=1&page=1` to get one result.

AUTHORIZATION: Bearer <token>

QUERY PARAMETERS

- `size` integer, nullable
  Default: `100`
  Minimum: `1`, Maximum: `1000`
  Specifies the maximum number of elements the endpoint will specify the maximum number of elements that the server is willing to process. Authorization service MAY ignore any value greater than 1000. Authorization service MAY reject requests with `size=0`. Clients SHOULD NOT MUST be prepared that the response contains more elements than the specified size. Clients SHOULD NOT MUST be prepared that the response contains less elements than the specified size. Authorization service MAY ignore any value greater than 1000.
- `page` integer, nullable
  Default: `1`
  Minimum: `1`
  Specifies the page number to be returned.

Pagination is OPTIONAL. Authorization service may not support it. Treating `?size=1&page=1` as an implementation is a way that clients can supporting pagination get one result. Authorization service MAY reject requests that are too complex to process MAY reject all requests.

If the response is paginated, the `Link` header MUST be used as defined in RFC 8288 to provide navigation URLs. Authorization service SHOULD NOT MUST omit `Link` header. Clients SHOULD NOT MUST be prepared that `Link` header is not provided. Clients SHOULD NOT MUST be prepared that `Link` header contains multiple URLs. Clients SHOULD NOT MUST be prepared that `Link` header contains URLs that are unknown or specified by the client. Authorization service SHOULD NOT MUST use the `Link` header to provide navigation URLs.

Response samples

200 Batch jobs

Content-type: application/batch-job+list

```
[
  {
    "id": "",
    "summary": "",
    "description": "",
    "processDefinition": "JSON-Graph-Process-Definition",
    "process": {
      "nodes": [
        {
          "id": "",
          "type": "activity",
          "parameters": {
            "activityId": "",
            "activityName": "",
            "timeout": 0,
            "retryPolicy": {
              "initialInterval": 0,
              "backoffCoefficient": 0,
              "maximumInterval": 0,
              "maximumAttempts": 0
            }
          },
          "metadata": {
            "additionalProperties": {}
          }
        }
      ],
      "edges": [
        {
          "id": "",
          "source": "",
          "sourceHandle": "success",
          "target": "",
          "metadata": {
            "additionalProperties": {}
          }
        }
      ],
      "additionalProperties": {}
    }
  }
]
```

Copy Expand all Collapse all

400 Bad request

Content-type: application/problem+json

```
{
  "type": "",
  "title": "",
  "status": 0,
  "detail": "",
  "instance": "",
  "errors": [
    {
      "pointer": "",
      "detail": ""
    }
  ]
}
```

Copy Expand all Collapse all

401 Unauthorized

Content-type: application/problem+json

```
{
  "type": "",
  "title": "",
  "status": 0,
  "detail": "",
  "instance": "",
  "errors": [
    {
      "pointer": "",
      "detail": ""
    }
  ]
}
```

Copy Expand all Collapse all

403 Forbidden

Content-type: application/problem+json

```
{
  "type": "",
  "title": "",
  "status": 0,
  "detail": "",
  "instance": "",
  "errors": [
    {
      "pointer": "",
      "detail": ""
    }
  ]
}
```

Copy Expand all Collapse all

500 Internal server error

Content-type: application/problem+json

```
{
  "type": "",
  "title": "",
  "status": 0,
  "detail": "",
  "instance": "",
  "errors": [
    {
      "pointer": "",
      "detail": ""
    }
  ]
}
```

Copy Expand all Collapse all

Create a new batch job

POST /v1/authorization/batch

Creates a new batch processing job from one or more (chained) processes or the best-cost

Processing the data does not start yet. The job stays jobs (chained) `scheduled` by default.

AUTHORIZATION: Bearer <token>

REQUEST BODY SCHEMA: application/batch-job-create

Specifies the job details, e.g. the user defined process and timing details.

- `id` string, nullable
  A short identifier for easing debugging and logs.
- `summary` string, nullable
  A short summary of what the process does. Optional.
- `description` string, nullable
  Detailed description to explain the workflow.
- `processDefinition` enum (required)
  Specifies the job details, e.g. the user defined process and timing details. Currently, the following values are accepted: `JSON-Graph-Process-Definition` (JSON Graph Format) and `Cadence-Flow` (Cadence Flow). Other values may be accepted in other implementations.
  Enum: `"JSON-Graph-Process-Definition"` `"Cadence-Flow"`
- `process` object (required)
  A JSON representation of a process graph.
  - `nodes` array of objects (required)
    A list of nodes.
    - `id` string (required)
      Unique identifier for this node.
    - `type` enum (required)
      Specifies the type of node. Currently, the following values are accepted: `activity` (Activity), `decision` (Decision), `parallel` (Parallel), `loop` (Loop), `start` (Start), `end` (End), `fail` (Fail). Other values may be accepted in other implementations.
      Enum: `"activity"` `"decision"` `"parallel"` `"loop"` `"start"` `"end"` `"fail"`
    - `parameters` object, nullable
      Defines the parameters which are passed to this node. Optional.
      - `activityId` string
        Specifies the activity to be executed.
      - `activityName` string
        Human readable name of the activity. Optional.
      - `timeout` integer, nullable
        Specifies the maximum time in seconds the activity may take to complete before it times out. Optional.
      - `retryPolicy` object, nullable
        Defines how the activity should be retried if it fails. Optional.
        - `initialInterval` integer
          Specifies the initial wait time in seconds before the first retry attempt.
        - `backoffCoefficient` number
          Specifies the multiplier used to calculate the wait time between retry attempts.
        - `maximumInterval` integer, nullable
          Specifies the maximum wait time in seconds between retry attempts. Optional.
        - `maximumAttempts` integer
          Specifies the maximum number of attempts. Zero means unlimited.
    - `metadata` object, nullable
      Optional metadata for UI rendering. Ignored by the workflow engine.
    - `additionalProperties` object
    - `additionalProperties` object
  - `edges` array of objects (required)
    A list of edges.
    - `id` string (required)
      Unique identifier for this edge.
    - `source` string (required)
      Specifies the source node id.
    - `sourceHandle` enum, nullable
      Specifies the source handle. Optional. Currently, the following values are accepted: `success` (Success), `failure` (Failure), `loop` (Loop). Other values may be accepted in other implementations.
      Enum: `"success"` `"failure"` `"loop"`
    - `target` string (required)
      Specifies the target node id.
    - `metadata` object, nullable
      Optional metadata for UI rendering. Ignored by the workflow engine.
    - `additionalProperties` object
    - `additionalProperties` object
  - `additionalProperties` object
  - `additionalProperties` object
- `additionalProperties` object
- `additionalProperties` object

Response samples

200 Batch job created

Content-type: application/batch-job

```
{
  "id": "",
  "summary": "",
  "description": "",
  "processDefinition": "JSON-Graph-Process-Definition",
  "process": {
    "nodes": [
      {
        "id": "",
        "type": "activity",
        "parameters": {
          "activityId": "",
          "activityName": "",
          "timeout": 0,
          "retryPolicy": {
            "initialInterval": 0,
            "backoffCoefficient": 0,
            "maximumInterval": 0,
            "maximumAttempts": 0
          }
        },
        "metadata": {
          "additionalProperties": {}
        }
      }
    ],
    "edges": [
      {
        "id": "",
        "source": "",
        "sourceHandle": "success",
        "target": "",
        "metadata": {
          "additionalProperties": {}
        }
      }
    ],
    "additionalProperties": {}
  }
}
```

Copy Expand all Collapse all

400 Bad request

Content-type: application/problem+json

```
{
  "type": "",
  "title": "",
  "status": 0,
  "detail": "",
  "instance": "",
  "errors": [
    {
      "pointer": "",
      "detail": ""
    }
  ]
}
```

Copy Expand all Collapse all

401 Unauthorized

Content-type: application/problem+json

```
{
  "type": "",
  "title": "",
  "status": 0,
  "detail": "",
  "instance": "",
  "errors": [
    {
      "pointer": "",
      "detail": ""
    }
  ]
}
```

Copy Expand all Collapse all

403 Forbidden

Content-type: application/problem+json

```
{
  "type": "",
  "title": "",
  "status": 0,
  "detail": "",
  "instance": "",
  "errors": [
    {
      "pointer": "",
      "detail": ""
    }
  ]
}
```

Copy Expand all Collapse all

500 Internal server error

Content-type: application/problem+json

```
{
  "type": "",
  "title": "",
  "status": 0,
  "detail": "",
  "instance": "",
  "errors": [
    {
      "pointer": "",
      "detail": ""
    }
  ]
}
```

Copy Expand all Collapse all

Modify a batch job

PATCH https://api.ebay.com/ sell/bulk_variant_update/v1_beta/inventory_item/{sku}/{locale}

Modifies an existing job in the field and inventory item identifier. Changes can be provided in the request payload.

Only one job per inventory item and locale can be in the queue at a time. Otherwise, requests to this endpoint doesn't be rejected with specific error codes (errorId/domain).

AUTHORIZATION	Bearer
OAUTH scopes required	none

URI_ID	string:sku
	Example: 24-inch-monitor
	Identifier of the inventory item.
URI_ID	string:locale
	Example: en-US,en-GB,de-DE,fr-FR
	Identifier of the locale.

REQUEST PAYLOAD application/ json

Specifies the job details to update.

locale	string:locale
	Specifies the locale of the existing downloaded variations.
	Example: en-US,en-GB,de-DE,fr-FR
	Supported locales: en-US, en-GB, de-DE, fr-FR, fr-BE, nl-BE, nl-NL, pl-PL
	Description: Locale filter description to update the variations.
	Locale enumeration type: See the enumeration type for representation.
price	object:price type:Amount
	Specifies the price to update the existing variations with previous marketplace.
	Example: "price": { "value": "19.99", "currency": "USD" }
	The following object describes the price.
currency	string:currency
	Specifies the currency for the price.
	Example: "currency": "USD"
	Enumeration type: See the enumeration type currency.
value	string:decimal
	Specifies the new price value.
	Example: "value": "19.99"
	Pattern: [0-9]+(\.[0-9]{1,3})
	Description: Decimal value.
	Decimal type: See the decimal type for more representation.
quantity	object:ship-to-quantity
	Specifies the ship-to-quantity to update the existing variations with previous marketplace.
	Example: "quantity": { "shipToQuantity": 10 }
	The following object describes the ship-to-quantity.
shipToQuantity	integer:int32
	Specifies the ship-to-quantity.
	Example: "shipToQuantity": 10
	Minimum: 0
	Description: Integer value.
	Integer type: See the integer type for more representation.
status	array:enum
	Specifies the status to update the existing downloaded variations.
	Example: "status": ["ACTIVE", "OUT_OF_STOCK"]
	Supported values: ACTIVE, OUT_OF_STOCK, PUBLISHED, UNPUBLISHED, DELETED
	Description: Array of status.
	Enumeration type: See the enumeration type for more representation.
	Array type: See the array type for more representation.
allowed_operations	array:enum
	Specifies the allowed operations to update the existing variations.
	Example: "allowed_operations": ["PUBLISH", "UNPUBLISH", "DELETE"]
	Supported values: PUBLISH, UNPUBLISH, DELETE
	Description: Array of allowed operations.
	Enumeration type: See the enumeration type for more representation.
	Array type: See the array type for more representation.
property_name	string:name
	Specifies the name of the property to update.
	Example: "property_name": "color"
	Maximum length: 40
	Description: Name of the property.
property_value	array:string
	Specifies the new property value to update the existing variations with previous marketplace.
	Example: "property_value": ["Red", "Blue", "Green"]
	Maximum length: 65
	Description: Array of property value.
	Array type: See the array type for more representation.

Response

HTTP STATUS CODE 204 Success

The request to modify a job was successful. The HTTP response code is 204 No Content, and the response body contains no data.

HTTP STATUS CODE 400 Bad Request

The request to modify a job failed. The HTTP response code is 400 Bad Request, and the response body contains a JSON error object. The following error codes can be returned in the error object:

Error Code	Long Message
20403	Invalid locale.
20404	Inventory item not found.
20405	Invalid inventory item.
20406	Invalid property name.
20407	Invalid property value.
20408	Invalid status.
20409	Invalid allowed operation.
20410	Invalid price.
20411	Invalid quantity.
20412	Invalid ship-to-quantity.
20413	Variations not found.
20414	Job not found.
20415	Job is in queue.
20416	Job is in progress.
20417	Job is completed.
20418	Job is cancelled.
20419	Job is failed.
20420	Job is partially completed.
20421	Job is expired.
20422	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20423	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20424	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20425	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20426	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20427	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20428	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20429	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20430	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20431	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20432	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20433	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20434	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20435	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20436	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20437	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20438	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20439	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20440	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20441	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20442	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20443	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20444	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20445	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20446	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20447	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20448	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20449	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20450	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20451	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20452	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20453	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20454	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20455	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20456	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20457	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20458	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20459	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20460	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20461	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20462	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20463	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20464	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20465	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20466	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20467	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20468	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20469	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20470	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20471	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20472	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20473	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20474	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20475	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20476	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20477	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20478	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20479	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20480	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20481	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20482	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20483	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20484	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20485	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20486	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20487	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20488	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20489	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20490	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20491	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20492	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20493	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20494	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20495	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20496	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20497	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20498	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20499	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20500	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20501	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20502	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20503	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20504	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20505	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20506	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20507	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20508	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20509	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20510	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20511	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20512	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20513	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20514	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20515	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20516	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20517	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20518	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20519	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20520	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20521	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20522	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20523	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20524	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20525	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20526	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20527	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20528	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20529	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20530	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20531	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20532	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20533	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20534	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20535	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20536	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20537	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20538	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20539	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20540	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20541	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20542	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20543	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20544	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20545	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20546	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20547	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20548	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20549	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20550	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20551	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20552	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20553	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20554	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20555	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20556	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20557	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20558	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20559	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20560	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20561	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20562	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20563	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20564	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20565	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20566	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20567	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20568	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20569	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20570	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20571	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20572	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20573	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20574	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20575	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20576	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20577	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20578	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20579	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20580	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20581	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20582	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20583	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20584	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20585	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20586	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20587	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20588	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20589	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20590	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20591	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20592	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20593	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20594	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20595	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20596	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20597	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20598	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20599	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20600	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20601	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20602	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20603	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20604	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20605	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20606	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20607	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20608	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20609	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20610	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20611	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20612	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20613	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20614	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20615	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20616	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20617	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20618	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20619	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20620	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20621	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20622	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20623	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20624	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20625	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20626	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20627	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20628	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20629	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20630	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20631	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20632	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20633	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20634	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20635	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20636	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20637	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20638	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20639	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20640	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20641	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20642	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20643	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20644	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20645	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20646	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20647	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20648	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20649	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20650	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20651	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20652	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20653	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20654	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20655	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20656	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20657	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20658	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20659	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20660	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20661	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20662	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20663	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20664	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20665	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20666	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20667	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20668	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20669	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20670	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20671	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20672	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20673	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20674	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20675	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20676	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20677	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20678	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20679	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20680	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20681	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20682	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20683	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20684	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20685	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20686	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20687	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20688	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20689	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20690	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20691	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20692	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20693	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20694	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20695	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20696	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20697	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20698	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20699	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20700	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20701	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20702	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20703	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20704	Job is completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20705	Job is cancelled. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20706	Job is failed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20707	Job is partially completed. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20708	Job is expired. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20709	Job is in queue. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20710	Job is in progress. Only one job per inventory item and locale can be in the queue at a time. Please wait for the job to be completed.
20711

Start processing a batch job

POST https://api.zai.zai/batches

Kick-off a job for the processing queue to compute the results.

The batch will be added to the formal queue shortly after the process. To specify the formal queue, please set `"processing_mode": "priority"`.

The job status can be queried. Processing results are available upon completion. Please check the job status in `/v1/batches/{batch_id}` and `/v1/files/{output_file_id}/content` after receiving the job and these endpoints report that the processing has ended.

Once processing starts, the status changes to `in_progress`.

- If the job completes successfully, the status changes to `completed`, and the processed results are available in the specified output file.
- If the job encounters a fatal error during processing, the status changes to `failed`.
- Whenever an error occurs during processing, the details `failed` will be `errors`.

The endpoint has an offset of 24 job statuses ahead, defined as `expiring`. In particular, it does not accept requests for jobs with a status of `expiring`, `expired`, `cancelling`, or `cancelled`.

Cancel a job in the processing queue if you no longer need the results. Please note that it takes time to obtain that the target is now in a fully processed state. Alternatively, check and GET the job status periodically. Upon receiving a cancellation request, the job status changes to `cancelling`. Once processed, they cannot retrieve/GET even users that requesting the target may be kept in a queue for a short period.

AUTHENTICATION ● Bearer

`job_id` string	字符长度 1-64 字符
	Example: `job-abcdefgh0123456789`
	Identifier of the batch job.

Response samples

200 **400** **401** **429** **500**

Content-type: application/json

```
{
  "id": "job-abcdefgh0123456789",
  "object": "batch",
  "endpoint": "/v1/chat/completions",
  "input_file_id": "file-abcdefgh0123456789",
  "completion_window": "24hours",
  "status": "in_progress",
  "output_file_id": null,
  "error_file_id": null,
  "created_at": 1722222222,
  "in_progress_at": 1722222222,
  "expiring_at": null,
  "finalizing_at": null,
  "completed_at": null,
  "failed": null,
  "expired": null,
  "cancelling": null,
  "cancelled": null,
  "request_counts": {
    "total": 0,
    "completed": 0,
    "failed": 0
  }
}
```

Copy Download

Cancel processing a batch job

POST https://api.zai.zai/batches/{batch_id}/cancel

Cancels an ongoing batch job. The job status changes to `cancelling`. Once processing stabilizes, it transitions to `cancelled`.

A request to cancel an expired batch job results in an error. Cancelling a completed batch job has no effect; the job remains in its completed state.

Upon receiving a cancellation request, partial results may still be available in the specified output file for the time it was canceled.

The endpoint has an offset of 24 job status ahead, defined as `expiring`.

Check the status in `/v1/batches/{batch_id}` to confirm cancellation. Alternatively, poll periodically or preferably monitor via webhook for notifications. Otherwise the status is set to `cancelled`.

AUTHENTICATION ● Bearer

`job_id` string	字符长度 1-64 字符
	Example: `job-abcdefgh0123456789`
	Identifier of the batch job.

Response samples

200 **400** **401** **429** **500**

Content-type: application/json

```
{
  "id": "job-abcdefgh0123456789",
  "object": "batch",
  "endpoint": "/v1/chat/completions",
  "input_file_id": "file-abcdefgh0123456789",
  "completion_window": "24hours",
  "status": "cancelling",
  "output_file_id": null,
  "error_file_id": null,
  "created_at": 1722222222,
  "in_progress_at": 1722222222,
  "expiring_at": null,
  "finalizing_at": null,
  "completed_at": null,
  "failed": null,
  "expired": null,
  "cancelling": null,
  "cancelled": null,
  "request_counts": {
    "total": 0,
    "completed": 0,
    "failed": 0
  }
}
```

Copy Download

Batch Jobs

Management of batch processing tasks (jobs) and their results.

List all batch jobs

GET https://api.zai.zai/batches

Lists all batch jobs submitted by a user.

It accepts **`BATCHSORDER`** to take the response order and by setting an optional `after` cursor and a `limit` for the number of batches to retrieve. The `after` cursor is an object ID that defines your place in the list when requesting subsequent pages.

AUTHENTICATION ● Bearer

QUERY PARAMETERS optional/object

`limit` integer	分页大小 默认值: `10` 最小值: `10` 最大值: `100`
	分页查询的最大结果数。
	默认值为 10，最小值为 10，最大值为 100。
`after` string	分页游标，用于获取指定对象之后的数据。
	通常设置为上一次查询结果中最后一条数据的 ID，以获取下一页数据。
`order` string	按创建时间排序的顺序。
	`asc` 表示升序，`desc` 表示降序。
	默认为 `desc`。

Response samples

200 **400** **401** **429** **500**

Content-type: application/json

```
{
  "object": "list",
  "data": [
    {
      "id": "job-abcdefgh0123456789",
      "object": "batch",
      "endpoint": "/v1/chat/completions",
      "input_file_id": "file-abcdefgh0123456789",
      "completion_window": "24hours",
      "status": "completed",
      "output_file_id": "file-abcdefgh0123456789",
      "error_file_id": null,
      "created_at": 1722222222,
      "in_progress_at": 1722222222,
      "expiring_at": null,
      "finalizing_at": 1722222222,
      "completed_at": 1722222222,
      "failed": null,
      "expired": null,
      "cancelling": null,
      "cancelled": null,
      "request_counts": {
        "total": 1,
        "completed": 1,
        "failed": 0
      }
    }
  ],
  "first_id": "job-abcdefgh0123456789",
  "last_id": "job-abcdefgh0123456789",
  "has_more": false
}
```

Copy Download

Create a new batch job

POST https://api.zai.zai/batches

Initiate a new batch processing task (job) from one or more (internal) processors at the local end.

Processing the data does not start yet. The job stores jobs information and `metadata` for refresh.

AUTHENTICATION ● Bearer

REQUEST BODY SCHEMA: application/json required

Specify the job details, e.g. the uploaded process and limiting details.

`endpoint` string	分组大小必须大于 1
	Enum: `"/v1/chat/completions"` `"/v4/chat/completions"` `"/v1/embeddings"`
	批处理操作的目标端点。
	目前支持 `/v1/chat/completions`、`/v4/chat/completions` 和 `/v1/embeddings`。
`input_file_id` string	上传文件的 ID，用于批处理请求。
	文件必须采用 JSONL 格式，每行代表一个请求，格式取决于端点。
	文件大小限制为 100 MB，最多可包含 50,000 个请求。
`auto_delete_input_files` boolean	默认值 false
	指示在处理完成后是否自动删除输入文件。
`processing_mode` string	处理模式，用于指定处理优先级。
	Enum: `"priority"` `"default"`
	`priority` 表示高优先级处理，`default` 表示标准处理。
`priority` integer	处理优先级，数值越大表示优先级越高。
	默认值为 0。
	仅在 `processing_mode` 为 `priority` 时有效。
`metadata` object	最多 16 个键值对
	可选的元数据，以键值对形式附加到对象。
	键最多 64 个字符，值最多 512 个字符。
`callback_uri` string	回调 URI，用于接收任务状态更新通知。
	当任务状态发生变化时，系统将向此 URI 发送 POST 请求。
	请确保该 URI 可公网访问并支持 HTTPS。
`callback_headers` object	回调请求时发送的自定义 HTTP 头部。
	格式为 JSON 对象，键为头部名称，值为头部值。
	例如：`{"Authorization": "Bearer token", "X-Custom": "value"}`
`expiration_seconds` integer	任务过期时间（秒）。
	从创建时间开始计算，超过此时间后任务将自动过期。
	最小值为 86400（1 天）。
`additional` object	扩展参数，用于传递特定模型或服务的额外配置。
	格式为 JSON 对象，具体内容取决于所选模型。
`sensitive` object	敏感参数，用于传递密钥、令牌等安全信息。
	格式为 JSON 对象，内容会在日志中被脱敏处理。

Response samples

200 **400** **401** **429** **500**

Content-type: application/json

```
{
  "id": "job-abcdefgh0123456789",
  "object": "batch",
  "endpoint": "/v1/chat/completions",
  "input_file_id": "file-abcdefgh0123456789",
  "completion_window": "24hours",
  "status": "validating",
  "output_file_id": null,
  "error_file_id": null,
  "created_at": 1722222222,
  "in_progress_at": null,
  "expiring_at": null,
  "finalizing_at": null,
  "completed_at": null,
  "failed": null,
  "expired": null,
  "cancelling": null,
  "cancelled": null,
  "request_counts": {
    "total": 0,
    "completed": 0,
    "failed": 0
  }
}
```

Copy Download

Modify a batch job

PUT https://api.zai.zai/batches/{batch_id}

Modify an existing job for the batch and save/restore the identifier. Changes can be processed in the queue, or not.

The job status does not change.

Jobs can only be modified if the job is not queued and running. Otherwise, requests to this endpoint MAY be rejected with a partial `errors` object.

AUTHENTICATION ● Bearer

`job_id` string	字符长度 1-64 字符
	Example: `job-abcdefgh0123456789`
	Identifier of the batch job.

REQUEST BODY SCHEMA: application/json required

Specify the job details to update.

`name` string	分组大小必须大于 1
	批处理任务的可读名称，用于标识和管理目的。
`description` string	分组大小必须大于 1
	批处理任务的可选描述，提供有关任务目的和内容的更多上下文。
`auto_delete_input_files` boolean	默认值 false
	指示在处理完成后是否自动删除输入文件。
`processing_mode` string	处理模式，用于指定处理优先级。
	Enum: `"priority"` `"default"`
	`priority` 表示高优先级处理，`default` 表示标准处理。
`priority` integer	处理优先级，数值越大表示优先级越高。
	默认值为 0。
	仅在 `processing_mode` 为 `priority` 时有效。
`metadata` object	最多 16 个键值对
	可选的元数据，以键值对形式附加到对象。
	键最多 64 个字符，值最多 512 个字符。
`callback_uri` string	回调 URI，用于接收任务状态更新通知。
	当任务状态发生变化时，系统将向此 URI 发送 POST 请求。
	请确保该 URI 可公网访问并支持 HTTPS。
`callback_headers` object	回调请求时发送的自定义 HTTP 头部。
	格式为 JSON 对象，键为头部名称，值为头部值。
	例如：`{"Authorization": "Bearer token", "X-Custom": "value"}`
`expiration_seconds` integer	任务过期时间（秒）。
	从创建时间开始计算，超过此时间后任务将自动过期。
	最小值为 86400（1 天）。
`additional` object	扩展参数，用于传递特定模型或服务的额外配置。
	格式为 JSON 对象，具体内容取决于所选模型。
`sensitive` object	敏感参数，用于传递密钥、令牌等安全信息。
	格式为 JSON 对象，内容会在日志中被脱敏处理。

Response samples

200 **400** **401** **429** **500**

Content-type: application/json

```
{
  "id": "job-abcdefgh0123456789",
  "object": "batch",
  "endpoint": "/v1/chat/completions",
  "input_file_id": "file-abcdefgh0123456789",
  "completion_window": "24hours",
  "status": "validating",
  "output_file_id": null,
  "error_file_id": null,
  "created_at": 1722222222,
  "in_progress_at": null,
  "expiring_at": null,
  "finalizing_at": null,
  "completed_at": null,
  "failed": null,
  "expired": null,
  "cancelling": null,
  "cancelled": null,
  "request_counts": {
    "total": 0,
    "completed": 0,
    "failed": 0
  }
}
```

Copy Download

Full metadata for a batch job

GET https://api.zai.zai/batches/{batch_id}

Lists all information about a submitted batch job.

AUTHENTICATION ● Bearer

`job_id` string	字符长度 1-64 字符
	Example: `job-abcdefgh0123456789`
	Identifier of the batch job.

Response samples

200 **400** **401** **429** **500**

Content-type: application/json

```
{
  "id": "job-abcdefgh0123456789",
  "object": "batch",
  "endpoint": "/v1/chat/completions",
  "input_file_id": "file-abcdefgh0123456789",
  "completion_window": "24hours",
  "status": "completed",
  "output_file_id": "file-abcdefgh0123456789",
  "error_file_id": null,
  "created_at": 1722222222,
  "in_progress_at": 1722222222,
  "expiring_at": null,
  "finalizing_at": 1722222222,
  "completed_at": 1722222222,
  "failed": null,
  "expired": null,
  "cancelling": null,
  "cancelled": null,
  "request_counts": {
    "total": 1,
    "completed": 1,
    "failed": 0
  }
}
```

Copy Download

Delete a batch job

DELETE https://api.zai.zai/batches/{batch_id}

Deletes all data related to a given batch job. Computations are stopped and computed results are deleted. You can still get partial results after deleting a batch job.

AUTHENTICATION ● Bearer

`job_id` string	字符长度 1-64 字符
	Example: `job-abcdefgh0123456789`
	Identifier of the batch job.

Response samples

200 **400** **401** **429** **500**

Content-type: application/json

```
{
  "id": "job-abcdefgh0123456789",
  "object": "batch",
  "deleted": true
}
```

Copy Download

Get an estimate for a batch job

`POST` /v3/job/{job_id}/estimate

Instead of a schedule for a batch job, that ends on decide or when calculation the quotes is ready, the user is a combination of them. Batch-end-processor (BEP) specify an expiry time for the quotes, after which the quotes are no longer valid. The value of BEP specifies the conditions for automatic ending. The conditions are the combination of the following: the number of quotes received, the price of the quotes, the time since the first quote was received, and the time since the last quote was received. The quotes are the first to arrive, the lowest price, or the most recent. If no quotes meet the conditions, the job is cancelled. The BEP can be set when the job is created, or later. The default BEP is "manual" (the user decides).

AUTHORIZATION `Bearer`

PATH PARAMETERS

job_id *required*	string,uuid Example: 123e4567-e89b-12d3-a456-426614174000 Identifier of the target job

Responses

200 Success, and will asynchronously processing and envelope signals, or failure as 4xx or 5xx (JSON)

application/json

Response Schema | Response Example

status *enum:success|error*	string Enum: "success", "error" The top-level indicator, "success" or "error".
data *object|null*	object|null Contains the estimate or quotes. The value MUST be null when the conditions for automatic ending are met. The consumer can be notified of the final state via a subsequent GET /v3/job/{job_id}/estimate.
↳ estimate	string Estimated duration for the operation. Duration MUST be specified as an ISO 8601 duration.
↳ quotes *array|null*	array|null Array of received signals/quotes, or the one of the generated from "decide" BEP, ordered by signal time.
↳ signal_id	string,uuid Example: 550e8400-e29b-41d4-a716-446655440000 Specifies that every field of the response MUST be interpreted as the value for this signal. Other fields MUST be ignored (extensible). Joins to the signal log.
↳ signal_time	string,date-time Example: 2026-03-15T14:30:00.000Z The time when the provisioner a unit. Formatted as a RFC 3339 date-time.

200 Success, and will asynchronously processing and envelope signals, or failure as 4xx or 5xx (JSON)

application/problem+json

Response Schema | Response Example

type	string,uri-reference Default: "about:blank" A URI identifying the problem type.
title	string A short, human-readable summary of the problem type.
status	integer The HTTP status code for this occurrence.
detail	string A human-readable explanation specific to this occurrence.
instance	string,uri-reference A URI identifying the specific occurrence.
code *required*	string Machine-readable error code.
errors	array|null Optional array of field-level validation errors.
↳ pointer *required*	string JSON Pointer (RFC 6901) to the offending field.
↳ reason *required*	string Human-readable explanation for this field.
↳ expected	string|null What was expected (e.g., "integer", "uuid", "enum").
↳ received	any|null What was received.
trace_id	string,uuid Example: 550e8400-e29b-41d4-a716-446655440000 Correlation ID for tracing. Matches the X-Trace-Id response header.

Logs for a batch job

`GET` /v3/job/{job_id}/log

Lists log entries for the batch job, usually for debugging purposes.

Both public and log-only attributes are returned. Entries are ordered by time, then by sequence. Use `cursor` to paginate through large logs.

If requested consecutively, it is RECOMMENDED that clients use the offset parameter to get only the new entries.

Entries marked `public` (BY OPTIONAL, default: `private`) are visible to PROSUMERS in the dashboard by their owners.

AUTHORIZATION `Bearer`

PATH PARAMETERS

job_id *required*	string,uuid Example: 123e4567-e89b-12d3-a456-426614174000 Identifier of the target job

QUERY PARAMETERS

limit	integer Default: 100 Maximum: 1000 The limit identifier (property `id`) of the log entry from the last known `id` received. Used for incremental polling. Only entries with `id` greater than this value are returned. If omitted or invalid, the server starts from the first entry.
cursor	string Opaque pagination cursor. Use the value from the previous response's `pagination.next_cursor` to fetch the next page. Takes precedence over `offset`.
offset	integer Skip the first N entries. Alternative to `cursor` for simple pagination. Ignored if `cursor` is provided.
level	string,enum Filter by minimum log level. Returns entries at this level or higher (e.g., `warn` returns `warn` and `error`).
source	string,enum Filter by log source. Multiple values can be comma-separated (e.g., `worker,system`).
since	string,date-time Filter by time. Returns only entries created after this timestamp.
fields	string,enum Comma-separated list of fields to include in the response. Reduces bandwidth for large payloads. Available fields: `id`, `time`, `level`, `source`, `message`, `attempt`, `metadata`. Example: `fields=id,time,level,message`

Responses

200 Success

application/json

Response Schema | Response Example

data	array Array of log entries.
↳ id	integer,int64 Example: 1 Unique identifier for this log entry. Monotonically increasing within a job. Use this as `cursor` for incremental polling.
↳ time	string,date-time Example: 2026-03-15T14:30:00.000Z Timestamp when the log entry was created. Formatted as RFC 3339 date-time.
↳ level	string,enum Enum: "debug", "info", "warn", "error" Log level. Ordinal order: debug < info < warn < error. Use `level` filter to query by minimum level.
↳ source	string,enum Enum: "worker", "system", "provisioner", "scheduler", "user" Log source. Multiple values can be comma-separated in the `source` filter.
↳ message	string Human-readable log message. Maximum 65,536 characters. Longer messages are truncated.
↳ attempt	integer,int32 Example: 3 Execution attempt number (1-indexed). Null for system-level logs not associated with a specific attempt.
↳ metadata	object|null Optional structured data. Maximum 16 KB. Null if not provided. Example: `{"duration_ms": 1234, "exit_code": 0}`
pagination	object Pagination metadata.
↳ next_cursor	string|null Cursor for the next page. Null if no more entries. Pass this value as `cursor` in the next request.
↳ has_more	boolean Default: false Example: true `true` if more entries exist after this page. Continue polling with `since` or `cursor`.
↳ total	integer,int64 Total number of entries matching the query. Useful for progress indication. Omitted if unknown.

List batch job results

`GET` /v3/job/{job_id}/results

Lists output file metadata for the completed batch job, ordered by `file_index`. Use `limit` and `offset` for pagination. Each result includes `id` and `file_index` for downloading via `/v3/job/{job_id}/results/{id}`. Only available when `job.output_mode` is `"batch"`.

The response includes `output_schema` if the job was created with a schema. Use this to validate output structure before processing. Jobs without a schema return `null`.

Clients SHOULD use `Accept: application/ndjson` to receive a newline-delimited JSON stream instead of this endpoint. The NDJSON stream includes `_type` fields for each record, enabling mixed-type parsing. Accept: `application/ndjson` returns a stream of result records, one per line. Accept: `application/json` returns this paginated JSON envelope.

Cursor pagination is not supported. Use `offset` and `limit` for pagination. Results are ordered by `file_index` then `id`, so offset pagination is stable.

AUTHORIZATION `Bearer`

PATH PARAMETERS

job_id *required*	string,uuid Example: 123e4567-e89b-12d3-a456-426614174000 Identifier of the target job

QUERY PARAMETERS

limit	integer,int32 Default: 100 Maximum: 1000 Maximum number of results to return. Capped at 1000.
offset	integer,int32 Default: 0 Number of results to skip. Use with `limit` for pagination.

Responses

200 Success

application/ndjson

Response Schema | Response Example

output_schema	object|null Optional JSON Schema describing the structure of `output` in each result. Null if no schema was provided when the job was created.
results *array*	array Array of result records, ordered by `file_index` then `id`.
↳ id	integer,int64 Example: 1 Unique identifier for this result. Use with `/v3/job/{job_id}/results/{id}` to download individual results.
↳ file_index	integer,int32 Example: 0 Zero-based index of the input file this result corresponds to. Matches `input_files[file_index]`.
↳ output	any The JSON output from agent for this individual document. Structure MUST conform to `output_schema` if provided. Can be any JSON value: object, array, string, number, boolean, or null.
↳ meta	object|null Optional metadata from agent. Processing time, model version, confidence scores, etc. Structure is agent-specific. Null if agent does not provide metadata.
↳ _type	string Default: "result" Example: "result" Discriminator for NDJSON streaming. Always `"result"` for this record type.
total	integer,int32 Example: 42 Total number of results. Use for pagination calculation.
limit	integer,int32 Example: 100 Maximum results per page. Echoes the `limit` query parameter.
offset	integer,int32 Example: 0 Number of results skipped. Echoes the `offset` query parameter.

Start processing a batch job

`POST` /v3/job/{job_id}/start

Batch-end-to-the-processing begins to compute the results.

The result will be stored in the format specified in the process. To specify the format use a process-specific `output_mode`.

The call returns `202 Accepted`. Processing is asynchronous and runs in the background. Poll `GET /v3/job/{job_id}` for `status`. Webhook notifications are sent when status changes to `completed`, `failed`, or `cancelled`.

Idempotent: Calling on a job already `processing` returns `200 OK` with current job state. Safe to retry on network timeouts.

- Once the processing starts the design is set to `locked`.
- Jobs with `input_mode: "stream"` cannot be started via this endpoint.
- Attempt to start a job in processing returns `409 Conflict` with `error.code: "already_processing"`.

The endpoint has no effect if the job status is already `processing` or `completed`. In particular, it does not restart a completed job. To reprocess, create a new job or use a batch-end-to-the-processing.

Bulk-ended: If the job is in `open` state, this transitions `open` → `processing`. Only the job owner (`user_id`) is allowed to start the batch-end-to-the-processing. Unauthorized users receive `404 Not Found` (not `403 Forbidden`) to prevent enumeration. Jobs in `open` state with `input_mode: "stream"` are rejected. Stream-mode jobs cannot use this endpoint to begin processing.

AUTHORIZATION `Bearer`

PATH PARAMETERS

job_id *required*	string,uuid Example: 123e4567-e89b-12d3-a456-426614174000 Identifier of the target job

Responses

202 The request is accepted, processing is in progress

application/problem+json

Response Schema | Response Example

type	string,uri-reference Default: "about:blank" A URI identifying the problem type.
title	string A short, human-readable summary of the problem type.
status	integer The HTTP status code for this occurrence.
detail	string A human-readable explanation specific to this occurrence.
instance	string,uri-reference A URI identifying the specific occurrence.
code *required*	string Machine-readable error code.
errors	array|null Optional array of field-level validation errors.
↳ pointer *required*	string JSON Pointer (RFC 6901) to the offending field.
↳ reason *required*	string Human-readable explanation for this field.
↳ expected	string|null What was expected (e.g., "integer", "uuid", "enum").
↳ received	any|null What was received.
trace_id	string,uuid Example: 550e8400-e29b-41d4-a716-446655440000 Correlation ID for tracing. Matches the X-Trace-Id response header.

Cancel processing a batch job

`POST` /v3/job/{job_id}/cancel

Cancels all related computations for the job at the top-level. A call also processing, partial-output results are discarded.

A partial-output processing results may be available for downloading depending on the state of the job at the time it was cancelled.

Idempotent: Calling on a job already `cancelled` or `completed` is safely accepted. Clients can retry without fear of side effects.

The endpoint may be an effect of the job status is `queued` or `processing`.

The call returns a `409 Conflict` if the job is `failed`. Failed jobs cannot be cancelled as a preliminary results are unavailable for recovery. Otherwise the status is `cancelled`.

AUTHORIZATION `Bearer`

PATH PARAMETERS

job_id *required*	string,uuid Example: 123e4567-e89b-12d3-a456-426614174000 Identifier of the target job

Responses

200 The request is accepted, processing is in progress

application/problem+json

Response Schema | Response Example

type	string,uri-reference Default: "about:blank" A URI identifying the problem type.
title	string A short, human-readable summary of the problem type.
status	integer The HTTP status code for this occurrence.
detail	string A human-readable explanation specific to this occurrence.
instance	string,uri-reference A URI identifying the specific occurrence.
code *required*	string Machine-readable error code.
errors	array|null Optional array of field-level validation errors.
↳ pointer *required*	string JSON Pointer (RFC 6901) to the offending field.
↳ reason *required*	string Human-readable explanation for this field.
↳ expected	string|null What was expected (e.g., "integer", "uuid", "enum").
↳ received	any|null What was received.
trace_id	string,uuid Example: 550e8400-e29b-41d4-a716-446655440000 Correlation ID for tracing. Matches the X-Trace-Id response header.

Secondary Services

Set forward-access to data using other web service protocols.

Supported secondary web service protocols

`GET` /v3/s/{name}

Lists supported secondary web service protocols such as WebDAV, SSH, SFTP, FTP, HTTP, Prometheus, or S3 Gateway. The response includes each protocol's capabilities, current status, and the supported configuration settings and supported access permissions.

Use this endpoint to discover which protocols are available for a given dataset or process and the available configuration options.

- The process parameters MAY(OR) be referenced both in `x-x-input-schema` and `x-x-output-schema`. Implementations SHOULD(OR) validate that referenced parameters exist in the process input or output schema.
- Protocols with `"enabled": false` are listed for discoverability but cannot be used until enabled.
- Use `/v3/s/{name}/{path}` to access the protocol-specific endpoint.
- Use `/v3/s/{name}/{path}/{id}` to access individual resources.

AUTHORIZATION `Bearer`

PATH PARAMETERS

name *required*	string Example: "dataset-001" Dataset or process identifier.

Responses

200 Success

application/ndjson

Response Schema | Response Example

protocols *array*	array Array of supported protocols.
↳ name *enum:webdav|ssh|sftp|ftp|http|prometheus|s3*	string,enum Enum: "webdav", "ssh", "sftp", "ftp", "http", "prometheus", "s3" Protocol identifier.
↳ enabled	boolean Default: false Example: true Whether the protocol is currently enabled. Disabled protocols cannot be used.
↳ path	string Example: "/s/dataset-001/webdav" Base path for this protocol. Use `/v3/s/{name}/{path}` for protocol-specific endpoints.
↳ capabilities	object Protocol capabilities.
↳ ↳ read	boolean Default: false Example: true Supports read operations.
↳ ↳ write	boolean Default: false Example: true Supports write operations.
↳ ↳ list	boolean Default: false Example: true Supports listing operations.
↳ ↳ delete	boolean Default: false Example: true Supports delete operations.
↳ ↳ mkdir	boolean Default: false Example: true Supports directory creation.
↳ ↳ rmdir	boolean Default: false Example: true Supports directory removal.
↳ ↳ rename	boolean Default: false Example: true Supports rename operations.
↳ ↳ copy	boolean Default: false Example: true Supports copy operations.
↳ ↳ symlink	boolean Default: false Example: true Supports symlink operations.
↳ ↳ chmod	boolean Default: false Example: true Supports permission modification.
↳ ↳ chown	boolean Default: false Example: true Supports ownership modification.
↳ ↳ touch	boolean Default: false Example: true Supports timestamp modification.
↳ ↳ quota	boolean Default: false Example: true Supports quota operations.
↳ ↳ search	boolean Default: false Example: true Supports search operations.
↳ ↳ checksum	boolean Default: false Example: true Supports checksum operations.
↳ ↳ lock	boolean Default: false Example: true Supports file locking.
↳ ↳ versioning	boolean Default: false Example: true Supports versioning.
↳ ↳ streaming	boolean Default: false Example: true Supports streaming.
↳ ↳ max_file_size	integer,int64 Example: 5368709120 Maximum file size in bytes. Null if unlimited.
↳ ↳ max_path_length	integer,int32 Example: 4096 Maximum path length in characters. Null if unlimited.
↳ ↳ case_sensitive	boolean Default: false Example: true Whether paths are case sensitive.
↳ ↳ unicode_paths	boolean Default: false Example: true Whether unicode characters in paths are supported.
↳ ↳ atomic_rename	boolean Default: false Example: true Whether rename operation is atomic.
↳ ↳ resumable_uploads	boolean Default: false Example: true Whether resumable uploads are supported.
↳ ↳ resumable_downloads	boolean Default: false Example: true Whether resumable downloads are supported.
↳ ↳ hard_links	boolean Default: false Example: true Whether hard links are supported.
↳ ↳ extended_attributes	boolean Default: false Example: true Whether extended attributes are supported.
↳ ↳ sparse_files	boolean Default: false Example: true Whether sparse files are supported.
↳ ↳ partial_write	boolean Default: false Example: true Whether partial write operations are supported.
↳ ↳ partial_read	boolean Default: false Example: true Whether partial read operations are supported.
↳ ↳ max_concurrent_ops	integer,int32 Example: 10 Maximum concurrent operations. Null if unlimited.
↳ ↳ rate_limit	boolean Default: false Example: true Whether rate limiting is applied.
↳ ↳ rate_limit_ops_per_sec	integer,int32 Example: 100 Rate limit operations per second. Null if unlimited.
↳ ↳ rate_limit_bytes_per_sec	integer,int64 Example: 104857600 Rate limit bytes per second. Null if unlimited.
↳ ↳ encryption	boolean Default: false Example: true Supports encryption.
↳ ↳ encryption_at_rest	boolean Default: false Example: true Supports encryption at rest.
↳ ↳ encryption_in_transit	boolean Default: false Example: true Supports encryption in transit.
↳ ↳ encryption_algorithms	array Supported encryption algorithms.
↳ ↳ ↳	string Example: "AES-256-GCM"
↳ ↳ compression	boolean Default: false Example: true Supports compression.
↳ ↳ compression_algorithms	array Supported compression algorithms.
↳ ↳ ↳	string Example: "zstd"
↳ ↳ max_compression_level	integer,int32 Example: 22 Maximum compression level. Null if unlimited.
↳ ↳ extensions	object Protocol-specific extensions. Keys are extension names, values are implementation-specific.
↳ ↳ ↳ webdav_version	string Example: "1.0" WebDAV version. Only present for WebDAV.
↳ ↳ ↳ ssh_version	string Example: "2.0" SSH version. Only present for SSH.
↳ ↳ ↳ sftp_version	integer,int32 Example: 3 SFTP version. Only present for SFTP.
↳ ↳ ↳ ftp_version	string Example: "RFC 959" FTP version. Only present for FTP.
↳ ↳ ↳ http_version	string Example: "1.1" HTTP version. Only present for HTTP.
↳ ↳ ↳ prometheus_version	string Example: "2.0" Prometheus version. Only present for Prometheus.
↳ ↳ ↳ s3_version	string Example: "v4" S3 version. Only present for S3.
↳ ↳ ↳ supported_features	array Supported features.
↳ ↳ ↳ ↳	string Example: "multipart_upload"
↳ ↳ ↳ vendor_extensions	object Vendor-specific extensions.
↳ ↳ ↳ ↳	any
↳ ↳ ↳ ↳	any
↳ ↳ ↳ ↳	any

Publish a new service

PUT https://www.zoho.com/crm/bigin/api/v2/services/publish

Publish a new service and set it as [ACTIVE](), [DEVELOPER](), [DEVELOPER_DELETED](), or [DISABLED]() for the specified organization.

The secondary and service DEVELOPER process the underlying data on demand (based on process prerequisites provided in the new field process_details). Set [process_details]() = false(default) or [true]() for example, for the upload backup service, respectively. The available process prerequisites are [DATA_MIGRATION]() and [DATA_PROCESSING]().

Set [force_publish]() = true (default) to publish the service, even if the specified dependencies on the service type and whether it supports changing those are met.

AUTHORIZATION	Bearer
X-CRM-ORG-ID (required)	60034000001
X-EXTENDED-APPS (optional)	appsdeployment

The body expects to create the service with the following parameters.

service_id (optional)	String,Long Int (Max: 19) Unique identifier for existing (deleted) entities.
[service_name]() (required)	String,Text (Max: 50) Service name.
[processed_data]() (optional)	Object (Key-Value pairs) (Max: 1000) A service-given, system-controlled JSR process metadata.
[enabled]() (optional)	Boolean Indicates whether a service supports actual data. All available services support data for retrieval, so, [GET /services/{id}]() returns [true]() for any service. Set to [false]() for non-data-supporting services.
[support_type]() (optional)	Enum Values: [ONLY_ONE]() [MANY]() Indicates whether a service can be created only once or multiple times.
[dependent]() (optional)	Boolean Indicates whether a service depends on other services for its functionality. Set to [true]() if the service requires other services to function. Set to [false]() if the service operates independently.
[dependent_service_ids]() (optional)	Array,Long Int (Max: 19) Array of configuration settings, i.e. the setting values supported by the service. Specify the supported configuration settings in [configurations]() in [GET /services/{id}/configurations]() or [GET /services/configurations]() APIs. The supported configuration settings are specific to the service. Refer to the documentation of the service, implementation details or any other service dependent configurations.
[configurations]() (optional)	Array,Long Int (Max: 19) Array of configuration settings, i.e. the setting values supported by the service. The following are possible values for the service type, [DATA_MIGRATION]() and [DATA_PROCESSING]() services.
[data_type]() (optional)	String,Text (Max: 255) Indicates the data type to process such data. All available services support data for retrieval, so, [GET /services/{id}]() returns [text]() for any service. Set to [application/zip]() for the backup and restore service.
[is_mandatory]() (optional)	Boolean Indicates whether a service needs data to function. Set to [true]() if the service requires data to function. Set to [false]() if the service does not require data to function.
[is_multi_select]() (optional)	Boolean Indicates whether a service supports multi-select. All available services support multi-select. Set to [true]() if the service supports multi-select. Set to [false]() if the service does not support multi-select.
[order]() (optional)	Integer (Max: 19) Indicates the order in which the service should be displayed in a process. The order specified here is used in the UI to display the service. Set to [null]() to use the default order.
[is_required]() (optional)	Boolean Indicates whether a service is required for the functionality to proceed. Set to [true]() if the service is required. Set to [false]() if the service is optional.
[is_multi_select]() (optional)	Boolean Indicates whether a service supports multi-select. Set to [true]() if the service supports multi-select. Set to [false]() if the service does not support multi-select.
[order]() (optional)	Integer (Max: 19) Indicates the order in which the service should be displayed in a process. The order specified here is used in the UI to display the service. Set to [null]() to use the default order.
[is_required]() (optional)	Boolean Indicates whether a service is required for the functionality to proceed. Set to [true]() if the service is required. Set to [false]() if the service is optional.
[data_type]() (optional)	String,Text (Max: 255) Indicates the data type to process such data. All available services support data for retrieval, so, [GET /services/{id}]() returns [text]() for any service. Set to [application/zip]() for the backup and restore service.
[is_mandatory]() (optional)	Boolean Indicates whether a service needs data to function. Set to [true]() if the service requires data to function. Set to [false]() if the service does not require data to function.
[process_details]() (optional)	Object (Key-Value pairs) (Max: 1000) A process-specific, system-controlled JSR process metadata.
[process_prerequisites]() (optional)	Object (Key-Value pairs) (Max: 1000) Indicates the prerequisites for the secondary and service DEVELOPER process. Set [process_prerequisites]() = [{"DATA_MIGRATION": true}]() to indicate that the secondary and service DEVELOPER process requires data migration. Set [process_prerequisites]() = [{"DATA_PROCESSING": true}]() to indicate that the secondary and service DEVELOPER process requires data processing.
[is_secondary_process]() (optional)	Boolean Indicates whether the service requires secondary process. Set to [true]() if the service requires secondary process. Set to [false]() if the service does not require secondary process.
[is_service_process]() (optional)	Boolean Indicates whether the service requires service process. Set to [true]() if the service requires service process. Set to [false]() if the service does not require service process

path
required

string

Examples:

- `borders/europe.geojson` - A path without special chars. It describes a file `europe.geojson` in a folder called `borders`.
- `europe/%C3%B6sterreich/m%C3%BCnster.shp` - A path with special chars. It describes a file `münster.shp` in folders called `europe` and `österreich`.

Path of the file, relative to the user's root directory. MAY include folders, but MUST not include relative references such as `.` and `..`.

Folder and file names in the path MUST be url-encoded. The path separator `/` and the file extension separator `.` MUST NOT be url-encoded.

The URL-encoding may be shown incorrectly in rendered versions due to OpenAPI 3 not supporting path parameters which contain slashes. This may also lead to OpenAPI validators not validating paths containing folders correctly.

Responses

200 A file from the workspace.

RESPONSE SCHEMA: application/octet-stream

string <binary>

4XX
The request can not be fulfilled due to an error on client-side, i.e. the request is invalid. The client SHOULD NOT repeat the request without modifications.

The response body SHOULD contain a JSON error object. MUST be any HTTP status code specified in RFC 7231. This request MUST respond with HTTP status codes 401 if authorization is required or 403 if the authorization failed or access is forbidden in general to the authenticated user. HTTP status code 404 SHOULD be used if the value of a path parameter is invalid.

See also:

- Error Handling in the API in general.
- Common Error Codes

5XX
The request can not be fulfilled due to an error at the back-end. The error is never the client's fault and therefore it is reasonable for the client to retry the exact same request that triggered this response.

The response body SHOULD contain a JSON error object. MUST be any HTTP status code specified in RFC 7231. See also:

- Error Handling in the API in general.
- Common Error Codes

```
    "message": "Parameter 'sample' is missing.
  - "links": [
      + { … }
    ]
}
```

Upload a file to the workspace

PUT `/files/{path}`

Uploads a new file to the given path or updates an existing file if a file at the path exists.

Folders are created once required by a file upload. Empty folders can not be created.

AUTHORIZATIONS: *Bearer*

PATH PARAMETERS

path
required

string

Examples:

- `borders/europe.geojson` - A path without special chars. It describes a file `europe.geojson` in a folder called `borders`.
- `europe/%C3%B6sterreich/m%C3%BCnster.shp` - A path with special chars. It describes a file `münster.shp` in folders called `europe` and `österreich`.

Path of the file, relative to the user's root directory. MAY include folders, but MUST not include relative references such as `.` and `..`.

Folder and file names in the path MUST be url-encoded. The path separator `/` and the file extension separator `.` MUST NOT be url-encoded.

The URL-encoding may be shown incorrectly in rendered versions due to OpenAPI 3 not supporting path parameters which contain slashes. This may also lead to OpenAPI validators not validating paths containing folders correctly.

REQUEST BODY SCHEMA: application/octet-stream
required

string <binary>

Responses

200 The file has been uploaded successfully.

RESPONSE SCHEMA: application/json

path
required

string `^[^/\:'"]|^ \:"]*$`
Path of the file, relative to the root directory of the user's server-side workspace. MUST NOT start with a slash `/` and MUST NOT be url-encoded.

The Windows-style path name component separator `\` is not supported, always use `/` instead.

Note: The pattern only specifies a minimal subset of invalid characters. The back-ends MAY enforce additional restrictions depending on their OS/environment.

size

integer
File size in bytes.

modified

string <date-time>
Date and time the file has lastly been modified, formatted as a RFC 3339 date-time.

4XX
The request can not be fulfilled due to an error on client-side, i.e. the request is invalid. The client SHOULD NOT repeat the request without modifications.

The response body SHOULD contain a JSON error object. MUST be any HTTP status code specified in RFC 7231. This request MUST respond with HTTP status codes 401 if authorization is required or 403 if the authorization failed or access is forbidden in general to the authenticated user. HTTP status code 404 SHOULD be used if the value of a path parameter is invalid.

See also:

- Error Handling in the API in general.
- Common Error Codes

5XX
The request can not be fulfilled due to an error at the back-end. The error is never the client's fault and therefore it is reasonable for the client to retry the exact same request that triggered this response.

The response body SHOULD contain a JSON error object. MUST be any HTTP status code specified in RFC 7231. See also:

- Error Handling in the API in general.
- Common Error Codes

Response samples

200 | 4XX | 5XX

Content type
application/json

Copy

```
{
    "path": "folder/file.txt",
    "size": 1024,
    "modified": "2018-01-03T10:55:29Z"
}
```

Delete a file from the workspace

DELETE `/files/{path}`

Deletes an existing user-uploaded file specified by its path. Resulting empty folders MUST be deleted automatically.

Back-ends MAY support deleting folders including its files and sub-folders. If not supported by the back-end a `FileOperationUnsupported` error MUST be sent as response.

AUTHORIZATIONS: *Bearer*

PATH PARAMETERS

path
required

string

Examples:

- `borders/europe.geojson` - A path without special chars. It describes a file `europe.geojson` in a folder called `borders`.
- `europe/%C3%B6sterreich/m%C3%BCnster.shp` - A path with special chars. It describes a file `münster.shp` in folders called `europe` and `österreich`.

Path of the file, relative to the user's root directory. MAY include folders, but MUST not include relative references such as `.` and `..`.

Folder and file names in the path MUST be url-encoded. The path separator `/` and the file extension separator `.` MUST NOT be url-encoded.

The URL-encoding may be shown incorrectly in rendered versions due to OpenAPI 3 not supporting path parameters which contain slashes. This may also lead to OpenAPI validators not validating paths containing folders correctly.

Responses

204 The file has been successfully deleted at the back-end.

4XX
The request can not be fulfilled due to an error on client-side, i.e. the request is invalid. The client SHOULD NOT repeat the request without modifications.

The response body SHOULD contain a JSON error object. MUST be any HTTP status code specified in RFC 7231. This request MUST respond with HTTP status codes 401 if authorization is required or 403 if the authorization failed or access is forbidden in general to the authenticated user. HTTP status code 404 SHOULD be used if the value of a path parameter is invalid.

See also:

- Error Handling in the API in general.
- Common Error Codes

5XX
The request can not be fulfilled due to an error at the back-end. The error is never the client's fault and therefore it is reasonable for the client to retry the exact same request that triggered this response.

The response body SHOULD contain a JSON error object. MUST be any HTTP status code specified in RFC 7231. See also:

- Error Handling in the API in general.
- Common Error Codes

Response samples

4XX | 5XX

Content type
application/json

Copy Expand all Collapse all

```
{
    "id": "550e8400-e29b-11d4-a716-44665544000
    "code": "SampleError",
    "message": "Parameter 'sample' is missing.
  - "links": [
      + { … }
    ]
}
```