Workflow Execution Service

Table of Contents

Chapter 1. Overview

1.1. Version information

Version : 1.0.0

1.2. URI scheme

BasePath : /ga4gh/wes/v1
Schemes : HTTPS

1.3. Consumes

- `application/json`

1.4. Produces

- `application/json`

Chapter 2. Executive Summary

The Workflow Execution Service (WES) API provides a standard way for users to submit workflow requests to workflow execution systems, and to monitor their execution. This API lets users run a single workflow (currently CWL or WDL formatted workflows, other types may be supported in the future) on multiple different platforms, clouds, and environments.

Key features of the API:

- can request that a workflow be run
- can pass parameters to that workflow (e.g. input files, cmdline arguments)
- can get information about running workflows (e.g. status, errors, output file locations)
- can cancel a running workflow

Chapter 3. Introduction

This document describes the WES API and provides details on the specific endpoints, request formats, and response. It is intended to provide key information for developers of WES-compatible services as well as clients that will call these WES services.

Use cases include:

- "Bring your code to the data": a researcher who has built their own custom analysis can submit it to run on a dataset owned by an external organization, instead of having to make a copy of the data
- Best-practices pipelines: a researcher who maintains their own controlled data environment can find useful workflows in a shared directory (e.g. Dockstore.org), and run them over their data

Chapter 4. Standards

The WES API specification is written in OpenAPI and embodies a RESTful service philosophy. It uses JSON in requests and responses and standard HTTP/HTTPS for information transport.

Chapter 5. Authorization & Authentication

Users must supply credentials that establish their identity and authorization in order to use a WES endpoint. We recommend that WES implementations use an OAuth2 bearer token, although they can choose other mechanisms if appropriate. WES callers can use the `auth_instructions_url` from the service-info endpoint to learn how to obtain and use a bearer token for a particular implementation.

The WES implementation is responsible for checking that a user is authorized to submit workflow run requests. The particular authorization policy is up to the WES implementer.

Systems like WES need to also address the ability to pass credentials with jobs for input and output access. In the current version of WES, the passing of credentials to authenticate and authorize access to inputs and outputs, as well as mandates about necessary file transfer protocols to support, are out of scope. However, parallel work on the Data Object Service is addressing ways to pass around access credentials with data object references, opening up the possibility that a future version of WES will provide concrete mechanisms for workflow runs to access data using credentials different than those used for WES. This is a work in progress and support of DOS in WES will be added in a future release of WES.

Chapter 6. Paths

6.1. Run a workflow.

```
POST /runs
```

6.1.1. Description

This endpoint creates a new workflow run and returns a `RunId` to monitor its progress.

The `workflow_attachment` array may be used to upload files that are required to execute the workflow, including the primary workflow, tools imported by the workflow, other files referenced by the workflow, or files which are part of the input. The implementation should stage these files to a temporary directory and execute the workflow from there. These parts must have a Content-Disposition header with a "filename" provided for each part. Filenames may include subdirectories, but must not include references to parent directories with '..' – implementations should guard against maliciously constructed filenames.

The `workflow_url` is either an absolute URL to a workflow file that is accessible by the WES endpoint, or a relative URL corresponding to one of the files attached using `workflow_attachment`.

The `workflow_params` JSON object specifies input parameters, such as input files. The exact format of the JSON object depends on the conventions of the workflow language being used. Input files should either be absolute URLs, or relative URLs corresponding to files uploaded using `workflow_attachment`. The WES endpoint must understand and be able to access URLs supplied in the input. This is implementation specific.

The `workflow_type` is the type of workflow language and must be "CWL" or "WDL" currently (or another alternative supported by this WES instance).

The `workflow_type_version` is the version of the workflow language submitted and must be one supported by this WES instance.

See the `RunRequest` documentation for details about other fields.

6.1.2. Parameters

Type	Name	Schema
FormData	**tags** *optional*	string (application/json)
FormData	**workflow_attachment** *optional*	< string (binary) > array
FormData	**workflow_engine_parameters** *optional*	string (application/json)
FormData	**workflow_params** *optional*	string (application/json)

Type	Name	Schema
FormData	**workflow_type** *optional*	string
FormData	**workflow_type_version** *optional*	string
FormData	**workflow_url** *optional*	string

6.1.3. Responses

HTTP Code	Description	Schema
200		RunId
400	The request is malformed.	ErrorResponse
401	The request is unauthorized.	ErrorResponse
403	The requester is not authorized to perform this action.	ErrorResponse
500	An unexpected error occurred.	ErrorResponse

6.1.4. Consumes

- `multipart/form-data`

6.1.5. Tags

- WorkflowExecutionService

6.2. List the workflow runs.

```
GET /runs
```

6.2.1. Description

This list should be provided in a stable ordering. (The actual ordering is implementation dependent.) When paging through the list, the client should not make assumptions about live updates, but should assume the contents of the list reflect the workflow list at the moment that the first page is requested. To monitor a specific workflow run, use GetRunStatus or GetRunLog.

6.2.2. Parameters

Type	Name	Description	Schema
Query	**page_size** *optional*	OPTIONAL The preferred number of workflow runs to return in a page. If not provided, the implementation should use a default page size. The implementation must not return more items than `page_size`, but it may return fewer. Clients should not assume that if fewer than `page_size` items are returned that all items have been returned. The availability of additional pages is indicated by the value of `next_page_token` in the response.	integer (int64)
Query	**page_token** *optional*	OPTIONAL Token to use to indicate where to start getting results. If unspecified, return the first page of results.	string

6.2.3. Responses

HTTP Code	Description	Schema
200		RunListResponse
400	The request is malformed.	ErrorResponse
401	The request is unauthorized.	ErrorResponse
403	The requester is not authorized to perform this action.	ErrorResponse
500	An unexpected error occurred.	ErrorResponse

6.2.4. Tags

- WorkflowExecutionService

6.3. Get detailed info about a workflow run.

```
GET /runs/{run_id}
```

6.3.1. Description

This endpoint provides detailed information about a given workflow run. The returned result has information about the outputs produced by this workflow (if available), a log object which allows the stderr and stdout to be retrieved, a log array so stderr/stdout for individual tasks can be retrieved, and the overall state of the workflow run (e.g. RUNNING, see the State section).

6.3.2. Parameters

Type	Name	Schema
Path	**run_id** *required*	string

6.3.3. Responses

HTTP Code	Description	Schema
200		RunLog
401	The request is unauthorized.	ErrorResponse
403	The requester is not authorized to perform this action.	ErrorResponse
404	The requested workflow run not found.	ErrorResponse
500	An unexpected error occurred.	ErrorResponse

6.3.4. Tags

- WorkflowExecutionService

6.4. Cancel a running workflow.

```
POST /runs/{run_id}/cancel
```

6.4.1. Parameters

Type	Name	Schema
Path	**run_id** *required*	string

6.4.2. Responses

HTTP Code	Description	Schema
200		RunId
401	The request is unauthorized.	ErrorResponse
403	The requester is not authorized to perform this action.	ErrorResponse
404	The requested workflow run wasn't found.	ErrorResponse
500	An unexpected error occurred.	ErrorResponse

6.4.3. Tags

- WorkflowExecutionService

6.5. Get quick status info about a workflow run.

```
GET /runs/{run_id}/status
```

6.5.1. Description

This provides an abbreviated (and likely fast depending on implementation) status of the running workflow, returning a simple result with the overall state of the workflow run (e.g. RUNNING, see the State section).

6.5.2. Parameters

Type	Name	Schema
Path	**run_id** *required*	string

6.5.3. Responses

HTTP Code	Description	Schema
200		RunStatus
401	The request is unauthorized.	ErrorResponse
403	The requester is not authorized to perform this action.	ErrorResponse
404	The requested workflow run wasn't found.	ErrorResponse
500	An unexpected error occurred.	ErrorResponse

6.5.4. Tags

- WorkflowExecutionService

6.6. Get information about Workflow Execution Service.

```
GET /service-info
```

6.6.1. Description

May include information related (but not limited to) the workflow descriptor formats, versions supported, the WES API versions supported, and information about general service availability.

6.6.2. Responses

HTTP Code	Description	Schema
200		ServiceInfo
400	The request is malformed.	ErrorResponse
401	The request is unauthorized.	ErrorResponse

HTTP Code	Description	Schema
403	The requester is not authorized to perform this action.	ErrorResponse
500	An unexpected error occurred.	ErrorResponse

6.6.3. Tags

- WorkflowExecutionService

Chapter 7. Definitions

7.1. DefaultWorkflowEngineParameter

A message that allows one to describe default parameters for a workflow engine.

Name	Description	Schema
default_value *optional*	The stringified version of the default parameter. e.g. "2.45".	string
name *optional*	The name of the parameter	string
type *optional*	Describes the type of the parameter, e.g. float.	string

7.2. ErrorResponse

An object that can optionally include information about the error.

Name	Description	Schema
msg *optional*	A detailed error message.	string
status_code *optional*	The integer representing the HTTP status code (e.g. 200, 404).	integer

7.3. Log

Log and other info

Name	Description	Schema
cmd *optional*	The command line that was executed	< string > array
end_time *optional*	When the command stopped executing (completed, failed, or cancelled), in ISO 8601 format "%Y-%m-%dT%H:%M:%SZ"	string
exit_code *optional*	Exit code of the program	integer (int32)
name *optional*	The task or workflow name	string
start_time *optional*	When the command started executing, in ISO 8601 format "%Y-%m-%dT%H:%M:%SZ"	string

Name	Description	Schema
stderr *optional*	A URL to retrieve standard error logs of the workflow run or task. This URL may change between status requests, or may not be available until the task or workflow has finished execution. Should be available using the same credentials used to access the WES endpoint.	string
stdout *optional*	A URL to retrieve standard output logs of the workflow run or task. This URL may change between status requests, or may not be available until the task or workflow has finished execution. Should be available using the same credentials used to access the WES endpoint.	string

7.4. RunId

Name	Description	Schema
run_id *optional*	workflow run ID	string

7.5. RunListResponse

The service will return a RunListResponse when receiving a successful RunListRequest.

Name	Description	Schema
next_page_token *optional*	A token which may be supplied as `page_token` in workflow run list request to get the next page of results. An empty string indicates there are no more items to return.	string
runs *optional*	A list of workflow runs that the service has executed or is executing. The list is filtered to only include runs that the caller has permission to see.	< RunStatus > array

7.6. RunLog

Name	Description	Schema
outputs *optional*	The outputs from the workflow run.	object
request *optional*	The original request message used to initiate this execution.	RunRequest
run_id *optional*	workflow run ID	string
run_log *optional*	The logs, and other key info like timing and exit code, for the overall run of this workflow.	Log
state *optional*	The state of the run e.g. RUNNING (see State)	State

Name	Description	Schema
task_logs *optional*	The logs, and other key info like timing and exit code, for each step in the workflow run.	< Log > array

7.7. RunRequest

To execute a workflow, send a run request including all the details needed to begin downloading and executing a given workflow.

Name	Description	Schema
tags *optional*	OPTIONAL A key-value map of arbitrary metadata outside the scope of `workflow_params` but useful to track with this run request	< string, string > map
workflow_engine_parameters *optional*	OPTIONAL Additional parameters can be sent to the workflow engine using this field. Default values for these parameters can be obtained using the ServiceInfo endpoint.	< string, string > map
workflow_params *optional*	REQUIRED The workflow run parameterizations (JSON encoded), including input and output file locations	object
workflow_type *optional*	REQUIRED The workflow descriptor type, must be "CWL" or "WDL" currently (or another alternative supported by this WES instance)	string
workflow_type_version *optional*	REQUIRED The workflow descriptor type version, must be one supported by this WES instance	string
workflow_url *optional*	REQUIRED The workflow CWL or WDL document. When `workflow_attachments` is used to attach files, the `workflow_url` may be a relative path to one of the attachments.	string

7.8. RunStatus

Small description of a workflow run, returned by server during listing

Name	Schema
run_id *required*	string
state *optional*	State

7.9. ServiceInfo

A message containing useful information about the running service, including supported versions

and default settings.

Name	Description	Schema
auth_instructions_url *optional*	A web page URL with human-readable instructions on how to get an authorization token for use with a specific WES endpoint.	string
contact_info_url *optional*	An email address URL (mailto:) or web page URL with contact information for the operator of a specific WES endpoint. Users of the endpoint should use this to report problems or security vulnerabilities.	string
default_workflow_engine_parameters *optional*	Each workflow engine can present additional parameters that can be sent to the workflow engine. This message will list the default values, and their types for each workflow engine.	< DefaultWorkflowEngineParameter > array
supported_filesystem_protocols *optional*	The filesystem protocols supported by this service, currently these may include common protocols using the terms 'http', 'https', 'sftp', 's3', 'gs', 'file', or 'synapse', but others are possible and the terms beyond these core protocols are currently not fixed. This section reports those protocols (either common or not) supported by this WES service.	< string > array
supported_wes_versions *optional*	The version(s) of the WES schema supported by this service	< string > array
system_state_counts *optional*	The system statistics, key is the statistic, value is the count of runs in that state. See the State enum for the possible keys.	< string, integer (int64) > map
tags *optional*	A key-value map of arbitrary, extended metadata outside the scope of the above but useful to report back	< string, string > map
workflow_engine_versions *optional*	The engine(s) used by this WES service, key is engine name (e.g. Cromwell) and value is version	< string, string > map
workflow_type_versions *optional*	A map with keys as the workflow format type name (currently only CWL and WDL are used although a service may support others) and value is a workflow_type_version object which simply contains an array of one or more version strings	< string, WorkflowTypeVersion > map

7.10. State

- UNKNOWN: The state of the task is unknown. This provides a safe default for messages where this field is missing, for example, so that a missing field does not accidentally imply that the state is QUEUED.
- QUEUED: The task is queued.
- INITIALIZING: The task has been assigned to a worker and is currently preparing to run. For example, the worker may be turning on, downloading input files, etc.

- RUNNING: The task is running. Input files are downloaded and the first Executor has been started.
- PAUSED: The task is paused. An implementation may have the ability to pause a task, but this is not required.
- COMPLETE: The task has completed running. Executors have exited without error and output files have been successfully uploaded.
- EXECUTOR_ERROR: The task encountered an error in one of the Executor processes. Generally, this means that an Executor exited with a non-zero exit code.
- SYSTEM_ERROR: The task was stopped due to a system error, but not from an Executor, for example an upload failed due to network issues, the worker's ran out of disk space, etc.
- CANCELED: The task was canceled by the user.
- CANCELING: The task was canceled by the user, and is in the process of stopping.

Type : enum (UNKNOWN, QUEUED, INITIALIZING, RUNNING, PAUSED, COMPLETE, EXECUTOR_ERROR, SYSTEM_ERROR, CANCELED, CANCELING)

7.11. WorkflowTypeVersion

Available workflow types supported by a given instance of the service.

Name	Description	Schema
workflow_type_version *optional*	an array of one or more acceptable types for the `workflow_type`	< string > array