Globus Automate Client

Globus

Jul 27, 2022

GETTING STARTED

This is the command line interface and Python package for working with Globus Automate, primarily Globus Flows, any service implementing the Globus Action Provider Interface, and Globus Queues.

CHAPTER

ONE

QUICK START

1.1 Installation

The Globus Automate CLI and SDK contains both a command-line tool `globus-automate` and a client library for interacting with Actions and Flows. It requires Python 3.6+. If a supported version of Python is not already installed on your system, see this Python installation guide.

For new users and users who want to primarily use the CLI interface into `Globus Automate`, we highly recommend installing the package using pipx:

```
pipx install globus-automate-client
```

For users interested in programmatic access to `Globus Automate`, the SDK interface will be required. Simply install the `globus-automate-client` library using your choice of package manager. For example, if using the pip package manager:

```
pip install globus-automate-client
```

1.2 Basic Usage

Once installed, run the CLI with the help flag to receive a summary of its functionality:

```
globus-automate --help
```

Or, if programmatic access to the Flows service is required, simply import the SDK in a Python script:

```
import globus_automate_client
help(globus_automate_client)
```

Note: Most operations available on the CLI are also available in the SDK. For users getting familiar with the Automate service, we recommend starting off with the CLI interface.

1.3 Authentication

All CLI and SDK operations invoking Globus Automate services require authentication.

When using the SDK, all operations require that authentication be provided in the form of Globus Authorizers.

Upon first interaction with any Globus Automate service via the CLI, a text prompt will appear directing the user to a web URL where they can proceed through an authentication process using Globus Auth to consent to the service they are using the CLI to interact with. This typically only needs to be done the first time a particular service is invoked. Subsequently, the cached authentication information will be used. Authentication information is cached in the file `~/.globus_automate_tokens.json`. It is recommended that this file be protected. The file is in `json` format with section names based on the `scope` and it may be edited to remove particular scopes if done with care.

If re-authentication or re-consent is needed, the entire authentication cache may be deleted with the following command:

```
globus-automate session logout
```

Or, to invalidate the authentication cache's contents first and then remove it:

```
globus-automate session revoke
```

Note: These commands will remove locally cached authentication data for all Globus Automate services.

1.4 Example Flows

To get started with authoring flows, you may find it helpful to reference some examples. The following flows are publicly visible and demonstrate some simple and common operations you may want to copy into your own flows:

- *"Move (Globus Example)"* (flow ID `9123c20b-61e0-46e8-9469-92c999b6b8f2` at the time of writing).
- *"2 Stage Transfer (Globus Example)"* (flow ID `79a4653f-f8da-43b6-a581-5d3b345ad575`).
- *"Transfer Set Permissions (Globus Example)"* (flow ID `cdcd6d1a-b1c3-4e0b-8d4c-f205c16bf80c`).

CHAPTER

TWO

GLOBUS AUTOMATE OVERVIEW

The Globus Automate platform provides tools and services which can be used to create reliable processes for research data management. The platform builds on the foundation of Globus capabilities such as Authorization and Data Transfer.

The Automate Platform introduces a few key concepts which may then be extended and combined to create custom processes solving particular research data management problems. These concepts are `Action Providers`, `Actions`, and `Flows`. Read on to learn about how `Flows` orchestrate `Action Providers` together in order to create `Actions` that perform the actual automation.

2.1 Use Cases

The key to the platform is enabling users to orchestrate multiple processing steps into a single workflow, or `Flow`. Some of these steps are provided by `Globus Automate` and others of which may be custom implementations supporting a specific need. Examples of these workflows might be:

- Automatically detect data output from scientific instruments which is then transferred, processed, and indexed.
- Provide a curated pipeline for description, annotation and publication of research datasets.
- Run data transfers on a recurring schedule.

2.2 Action Providers

An `Action Provider` is an HTTP accessible service which acts as a single step in a process and implements the `Action Provider Interface`. When an `Action Provider` is invoked, it creates (or "provides") an `Action` which represents a single unit of work. Examples of units of work are running a file transfer using `Globus Transfer` or ingesting data into `Globus Search`.

Each `Action Provider` expects to be invoked with parameters particular to the service it provides. To support usability and discovery, each can be introspected to determine what its `input schema` or input properties are. Introspection also provides information such as who operates the `Action Provider`, descriptive text on the service it provides, and who can use the service. Access to `Action Providers` and their invocation is controlled via `Globus Auth`. Some of these services may be *synchronous* meaning that an invocation will complete in the context of the HTTP request that triggered it. Other services support *asynchronous* activities, meaning that the invocation will persist beyond the HTTP request that invoked it and the caller must monitor the `Action` for updates on when it is completed and its result.

Globus operates a series of these `Action Providers` available for public use. For a full list of these `Action Providers`, see section Globus Action Providers. Globus also supports users writing their own `Action Providers` via the Globus Action Provider Toolkit - a Python SDK that makes it easy to provide custom services that can be tied into the `Globus Automate` ecosystem of services.

These Action Providers form the foundation of Globus Automate and are primarily used by referencing their URLs in *Flows*. Globus Automate allows users to flexibly piece together these individual services to create reliable high level workflows.

2.3 Actions

An Action represents a single, discrete invocation of an Action Provider. It is record of an operation and includes details for its result, its current execution status, and metadata dictating which Globus Auth identities are allowed to read or modify the Action's state. Globus Automate services allow orchestrating these individual Actions into robust processes that can tolerate their distinct execution states, including success and failure. Users will not often need to operate on Actions directly, rather, the User will create a Run of a Flow and the Run will invoke Action Providers, creating Actions as necessary to accomplish the automation.

2.4 Flows

A Flow represents a single process that orchestrates a series of services into a self contained operation. One can think of a Flow as a declaratively defined ordering of Action Providers with condition handling to define expected success or failure scenarios.

A Flow may be defined and deployed to the Flows service by any user. When deploying, the user may control which other users can discover the Flow and separately, which users can run the Flow. All access control is provided by Globus Auth. Thus, Flows can easily and safely be shared among users. Once deployed, the Flow will receive a HTTP-accessible Flow URL which makes it available for use in Globus Automate.

It may also be interesting to note that once deployed, the Flow will implement the Action Provider Interface. What this means is that a Flow is technically a form of Action Provider, and as such it can be referenced by other Flows by its Flow URL. This allows for modularity in defining Flows and in a separation of concerns where SubFlows can be trusted to provide some process or behavior.

When users run an instance of the Flow, we call that a Run. A Run shares the Action interface, supporting operations such as viewing its status, cancelling its execution, and removing its execution state. This allows for common tooling and terminology for working with Runs and Actions. In general, any operation available on an Action will be possible on a Run and vice versa.

Globus Automate imposes no restrictions on how long a Run may execute or on the number of units of work defined in a Flow. We support long running Runs by providing support for monitoring and status updates.

2.5 Authentication and Authorization

An important consideration in the Globus Automate platform is authentication and authorization. All interactions with Action Providers Actions and Flows are authenticated by Globus Auth.

For Action Providers and Flows, authorization lists exist to control which identities can view its details and which identities can invoke an Action or Run of a Flow in the service. For Actions and Runs, authorization lists exist to enforce which identities may view its execution details and which identities may modify the execution status. For these authorization lists, identities may be specified as individual Globus users or as Globus Groups. Thus, while the Globus Automate platform is open to all users with access, it is possible to carefully control which users have access to your resources.

The authorization lists for a Flow definition are as follows:

- `flow_viewers`: Users who are allowed to see that the Flow is present in the system. Users not in the list will have no way of becoming aware that the Flow exists.
- `flow_starters`: Users who are allowed to start a particular Flow running. If not present in this list, a user attempting to start a Flow will receive an error.
- `flow_administrators`: Users who can administer a flow, most notably updating the Flow's execution instructions (its "definition") or other meta-data about the Flow.
- `flow_owner`: A single user who is considered the user with primary responsibility for maintaining the Flow. Users in the `flow_administrators` list may transfer ownership to themselves via a Flow update operation.

Once a Flow has been started, a "run" object is created for monitoring and managing the particular run. Authorization lists are created to determine which users may perform these operations as follows:

- `run_monitors`: These users may request the current state of the Flow run seeing what steps of the Flow have been executed, what the inputs and outputs of the various steps have been and see whether it has completed or not.
- `run_managers`: These users may also perform operations which alter the run of the Flow or its state. In particular, they may request that the run be canceled or they may remove the state for a completed run.
- `run_owner`: This is the user who initiated the run of the Flow. Unlike a `flow_owner` this role cannot be transferred to another user. The `run_owner` can perform the same operations as a user in the `run_managers` list.

For both Flows and Flow runs, the authorization lists are "cumulative" in the sense that a user in a particular list (termed having that "role") may also perform all the operations of users in the roles listed prior to it in the list. Thus, for example, a user in the `flow_administrators` list can perform all the operations associated with those in the `flow_viewers` and the `flow_starters` lists. Similarly, a user in `run_managers` can do all that those in `run_monitors` can.

Values within the authorization lists take the form of urns specifying individual users or groups of users based on Globus Groups. When specifying a user in an authorization list, the principal value will be the user's UUID prefixed with `urn:globus:auth:identity:`. When specifying a Globus Group in the list, the principal value needs to be the Group's UUID prefixed with `urn:globus:groups:id:`.

Tip: To determine a Globus user's ID, you can use the *globus* CLI:

```
globus get-identities username@globus.org
```

To determine the Globus Group's ID, you can search for the Group in the Globus Web Application.

Two special values, `public` and `all_authenticated_users` may also be used in some authorization lists. `public` indicates that the operation is allowed for requests that have no authorization and may be used in the `flow_viewers` list, and `all_authenticated_users` indicates that any user who presents a `Globus Auth` credential in the form of an access token is permitted access and may be used in a `flow_starters` list.

CHAPTER

THREE

USING THE CLI

For many, the primary way of interacting with Globus Automate services will be via the CLI.

3.1 Configuration

It is recommended to install `globus-automate` command line completion to support use and discovery of the available commands and options. Currently only the `bash`, `fish`, and `zsh` shells are supported.

```
globus-automate --install-completion
```

Note: For users of `zsh`, make sure the `zsh` completion system is correctly initialized by running `compinit` near the end of your `.zshrc` file.

3.2 General Usage

Each command support a `--verbose` option which can be used to see the underlying HTTP request information for what each command is doing. This can be useful for reusing tokens, debugging purposes, or for getting more familiar with the Globus Automate services' APIs.

Each command also supports a `--help` option which provides concise information on the command and documents its expected inputs.

In almost all cases, the output from each command will be in JSON format. The CLI will format the output to try to improve readability, however, you may wish to filter the output using pipes and other command line tools such as `grep` or `jq`.

Many of the commands involve specifying JSON formatted input. Anywhere a command makes use of JSON, it can be specified in one of two ways:

- **Directly on the commmand line**
  Typically by using single quote characters surrounding the content to protect it from interpretation by the shell:

  ```
  --option '{"property": "Value"}'
  ```

  where `--option` is the option requiring a JSON formatted parameter.
- **In a file**
  Place the JSON object in a file and refer to the file instead:

```
--option /path/to/your/input_file.json
```

where `--option` is the option requiring a JSON formatted parameter.

Commands which can run or monitor an Action or Flow Run support a `--watch` flag which can be used to stream updates on an activity's execution state.

Many operations require a principal ID to delegate access. This value represents a single Globus user identity or a Globus Group. The format for a principal is the user ID or Group ID, prefixed with `urn:globus:auth:identity:` if it is a user, or `urn:globus:groups:id:` if it is a Group.

3.3 Using the CLI with Actions

In Globus Automate, Actions represent a single unit of work. Actions are created by invoking `Action Providers`. The CLI provides a means of interacting with `Action Providers` to create and manage Actions, however in many cases, the fine grain nature of the Actions means they are not often interacted with directly.

To interact with an Action, you must know the URL for the `Action Provider` that created (or will create) it. This URL is specified using the `--action-url` option to the `action` subcommands.

Wherever an `--action-url` option is present, the `--action-scope` option may also be provided. The *scope string* is the Globus Auth scope registered for interacting with this Action Provider. This *scope string* is published as part of the `Action Provider`'s description (see *Introspection*) and the CLI will automatically retrieve this value when it is not specified by the user. In some cases, even retrieving the *scope string* may require authentication, and in these cases, introspecting the `Action Provider` is not possible without providing the `--action-scope` option.

All of the `Action Providers` operated by the Globus team are described in the section *Globus Action Providers*, which includes their URL and tips on using the CLI for interactions with these `Action Providers` directly. As these `Action Providers` are publicly viewable, there is no need to provide the `--action-scope` option when working with them from the CLI – the CLI will look up the *scope string* automatically.

As an example, we will work through the operations on the `Hello World Action Provider` at the URL `https://actions.globus.org/hello_world`.

3.3.1 Introspection

The first step to learning more about an `Action Provider` is using the introspect operation to get a description of the `Action Provider`:

```
globus-automate action introspect --action-url https://actions.globus.org/hello_world
```


```
{
    "admin_contact": "support@globus.org",
    "administered_by": [],
    "api_version": "1.0",
    "description": null,
    "event_types": null,
    "globus_auth_scope": "https://auth.globus.org/scopes/actions.globus.org/hello_world",
    "input_schema": {
        "additionalProperties": false,
        "properties": {
        "echo_string": {
            "type": "string"
```


(continues on next page)

(continued from previous page)


```
        },
        "required_dependent_scope": {
            "type": "string"
        },
        "sleep_time": {
            "type": "integer"
        }
        },
        "type": "object"
    },
    "keywords": null,
    "log_supported": false,
    "maximum_deadline": "P30D",
    "runnable_by": [
        "all_authenticated_users"
    ],
    "subtitle": "An Action responding Hello to an input value",
    "synchronous": false,
    "title": "Hello World",
    "types": [
        "ACTION"
    ],
    "visible_to": [
        "public"
    ]
}
```


From this introspection response we can see that the *scope string* for this `Action Provider` is the value of the `globus_auth_scope` field, `https://auth.globus.org/scopes/actions.globus.org/hello_world`. We can also see that the `admin_contact` is Globus.

For information on what this `Action Provider` does, it is useful to examine the `title`, `subtitle`, and `description` fields. We can also see that the `Action Provider` is `visible_to` *public*, meaning that anyone can make unauthenticated requests to the introspection endpoint. Similarly, it is `runnable_by` *all_authenticated_users*, meaning that any user with valid Globus Auth credentials may use this `Action Provider` to create Actions.

The most important information for our next step is the `input_schema` element as it provides a description of the input we need to form for running an Action on this `Action Provider`. The `input_schema` element is in JSON Schema format. This schema defines three properties: `echo_string`, `sleep_time`, and `required_dependent_scope`. We will use this information in the next section on running an Action.

3.3.2 Running

The first step to prepare for running an Action is to create a file containing the input to the Action. We'll call the file `hello_input.json` and it contains the following:


```
{
  "echo_string": "Welcome to Globus Automate!",
  "sleep_time": 60
}
```


This input conforms to the `input_schema` from the *Introspection* call, and specifies that we will have the Action echo a message back to us and that it will "sleep" for 60 seconds until the Action is complete. We'll use this sleep time to

demonstrate monitoring the state of an Action below.

With our input in place, run the Action using the following command:

```
globus-automate action run --action-url https://actions.globus.org/hello_world --body␣
↪hello_input.json
```

Note: If this is your first time running the `Hello World Action Provider` you will see text and a prompt appear on your terminal window. Follow the instructions to authenticate to Globus Auth to run this Action. This will only appear on the first time you interact with an `Action Provider`.

The resulting output will look like:


```
{
    "action_id": "CBOXB3fUdKrO",
    "completion_time": null,
    "creator_id": "urn:globus:auth:identity:06a24bef-940e-418a-97bc-48229c64cc99",
    "details": {
        "Hello": "World",
        "hello": "Welcome to Globus Automate!"
    },
    "display_status": "ACTIVE",
    "label": null,
    "manage_by": [
        "urn:globus:auth:identity:6f8c1345-33c6-4235-86c6-90fbadbf4d35",
        "urn:globus:auth:identity:06a24bef-940e-418a-97bc-48229c64cc99"
    ],
    "monitor_by": [
        "urn:globus:auth:identity:6f8c1345-33c6-4235-86c6-90fbadbf4d35",
        "urn:globus:auth:identity:06a24bef-940e-418a-97bc-48229c64cc99"
    ],
    "release_after": null,
    "start_time": "2021-04-29 23:21:47.763653+00:00",
    "status": "ACTIVE"
}
```


This output is referred to as an `Action Status` document and all output from working with Actions will follow this format.

The `action_id` is an identifier associated with this `Action Provider` invocation and is used to track this Action's lifecycle.

The `status` value of `ACTIVE` indicates that the Action is in the process of executing. The possible values for `status` are:

- **`ACTIVE`**
  The Action is running and making progress towards completion.
- **`INACTIVE`**
  The Action has not yet completed and it is not making progress. Commonly, some intervention is necessary to help it continue to make progress.
- **`SUCCEEDED`**
  The Action is complete and the completion was considered to be normal.
- **`FAILED`**

The Action has stopped running due to some error condition. It cannot make progress towards a successful completion.

Each Action can be provided a `label` to help identity the purpose for which it was run.

The `details` field format is specific to every `Action Provider` and is the output or result of running the Action. It will often contain information about why an Action has reached the state it is in.

The `release_after` field is an ISO8601 format time duration value that indicates how long after completion the `Action Provider` will retain a record of the Action's execution. Until then, the record will persist and can be looked up.

`monitor_by` represents delegated read-only access to the Action's execution state, meaning that principals in an Action's `monitor_by` field will be able to retrieve the Action's execution state (see *Retrieving Status*). Principals may be either a Globus Auth user or a Globus Auth group. The format for a Globus Auth user is `urn:globus:auth:identity:<UUID>` and for a Globus Auth group is `urn:globus:groups:id:<UUID>`.

`manage_by` represents delegated write access to the Action's execution state, meaning that principals in an Action's `manage_by` field will have the ability to change the alter the state it is in (see *Canceling and Releasing*). Principals may be either a Globus Auth user or a Globus Auth group. The format for a Globus Auth user is `urn:globus:auth:identity:<UUID>` and for a Globus Auth group is `urn:globus:groups:id:<UUID>`.

Since the Action has already been run, we cannot change any of these fields. If we wanted to run another Action with updated values for any of the fields, we would pass those as command line options. For information on how to use the options, run the command with `--help`:

```
globus-automate action run --help
```

Tip

You can specify each of the `--monitor-by` and `--manage-by` flags multiple times to provide multiple principals with read or write access on the Action.

3.3.3 Retrieving Status

Once an Action has been run, the user who initiated the Action or anyone in the Action's `monitor_by` field can monitor or retrieve its status as follows:

```
globus-automate action status --action-url https://actions.globus.org/hello_world
↪<action_id>
```

where the `action_id` is the value returned from the `action run` command from above. The output will be an Action Status document. When the Action is completed, the `completion_time` field will be present indicating when the Action reached its final state. You can continue requesting the Action's status as long as the Action exists on the `Action Provider`.

In out example, we asked the Action to "sleep" for 60 seconds. Therefore, the Action will remain in an `ACTIVE` state until 60 seconds have passed, at which point the status should be `SUCCEEDED`.

3.3.4 Canceling and Releasing

An Action which is running, but which is no longer needed, may be canceled (or released) by the user who initiated the Action execution or anyone in the Action's `manage_by` field using a command of the form:

```
globus-automate action cancel --action-url https://actions.globus.org/hello_world
↪<action_id>
```

The cancel operation is considered to be an advisory request from the user. Actions may not be cancelled immediately, or they may not be canceled at all. A request to cancel an Action which has reached a final state of either `SUCCEEDED` or `FAILED` will result in an error return.

To remove an Action's state from the `Action Provider`, the user who initiated the Action execution or anyone in the Action's `manage_by` field can use the release subcommand:

```
globus-automate action release --action-url https://actions.globus.org/hello_world
↪<action_id>
```

Release may only be performed on Actions which have reached a final state. If the Action is in either the `ACTIVE` or `INACTIVE` state, the release will fail.

Once released, the Action state is forever removed from the `Action Provider` and all attempts to access it will fail. `Action Providers` use the `maximum_deadline` field to advertise how long they will keep a record of an Action after it reaches a completed state. The time at which this will happen is equal to the `completion_time` plus the `release_after` values in the Action Status document.

3.4 Using the CLI with Flows

As described in the section on *Flows*, a Flow combines Actions and other operations into a more complex operation. When a Flow is invoked, it creates a `Run` and the `Run`'s interface is very much like an Action's; it has `run`, `status`, `cancel` and `release` operations defined. Because of this similarity, we sometimes refer to `Run`'s as Actions in the documentation, CLI and SDK.

The CLI contains commands for creating, defining, and managing Flows definition and commands for running, monitoring, and managing Flow `Runs` (also known as `Actions`).

Note: This section does not provide details on writing Flows. That is covered in greater detail in the section on *Authoring Flows for the Globus Flows Service*.

3.4.1 Finding and Displaying Flows

When a Flow is deployed to Automate, the creator can specify which identities the Flow should be visible to and which identities the Flow should be runnable by. As the names suggest, users in a Flow's `visible_to` field will be able to query the service to view a Flow's definition and metadata. Users in a Flow's `runnable_by` field will be able to run an instance of the Flow.

The following command will list the Flows you have created:

```
globus-automate flow list
```

To view Flows which are visible or runnable by you as well, run the following command:

```
globus-automate flow list --role created_by --role visible_to --role runnable_by
```

This outputs a list of Flows, where the description of each flow carries the same fields as the output from `globus-automate action introspect` described above. This emphasizes again the similarity between Flows and Actions. The `title` and `description` fields may be helpful in determining what a Flow does and what its purpose is. Like Actions, the `input_schema` may define what is required of the input when running the flow. However, not all Flows are required to define an `input_schema` as a convenience to Flow authors who may not be familiar with creating JSON Schema specifications. Importantly, each entry in the list of Flows will also contain a value for `id` which we refer to as the "Flow id" and denote as `flow_id` below. This value will be used for further interacting with a particular Flow.

To display information about a single Flow you may use:

```
globus-automate flow display <flow_id>
```

Or, to visualize the Flow:

```
globus-automate flow display <flow_id> --format image
```

When focusing on one Flow, it is also useful to notice the field `definition`. This is the actual encoding of the Flow as it was created and deployed by the Flow's author. Looking at this value may give further information about how the Flow works. This can be useful both to determine if a Flow performs the function you desire, but also as a method to see how other Flows have been defined if you are interested in creating new Flows.

3.4.2 Executing and Monitoring Flows

Execution and monitoring of Flows follows the same pattern as Actions: the run/status/cancel/release pattern is the same.

When initiating a Flow run, you can delegate access to the Flow instance to other Globus Auth identities. By providing the `monitor-by` option, you can delegate read-only access to other users or groups, allowing them to retrieve it execution state. By providing the `manage-by` option, you delegate write access to other users or groups, allowing them to alter its execution state. In the example below, we show how to run an instance of a Flow and delegate monitor access to a Globus Group:

```
globus-automate flow run <flow_id> --flow-input input.json \
    --monitor-by urn:globus:groups:id:00000000-0000-0000-0000-000000000000
```

Note: If no `manage_by` or `monitor_by` values are specified, only the identity instantiating the Flow run is allowed to monitor or manage a Flow's running state.

This acts like `globus-automate action run` with the flow id rather than the `action_url` specifying the "name" of the Action to be run. The output, like for Actions, will be an Action status document including an `action_id` which is used in the following commands:

```
globus-automate flow action-status --flow-id <flow_id> <action_id>
```

```
globus-automate flow action-cancel --flow-id <flow_id> <action_id>
```

```
globus-automate flow action-release --flow-id <flow_id> <action_id>
```

For each of these, the `details` provides information about the most recent, potentially final, state executed by the Flow. However, as the Flow may execute many states, it is useful to be able to see what states have been executed and what their input and output have been. This can be seen via the "log" of the Flow execution as follows:

```
globus-automate flow action-log --flow-id <flow_id> <action_id>
```

The log may have a large number of entries. You can request more entries be returned using the option `-limit N` where `N` is the number of log entries to return. The default value is 10.

3.4.3 Creating and managing Flows

Many users will only ever use Flows created by others, so they may not necessarily need to understand how to create Flows including the commands listed in this section. For those that have created a Flow, the first step is to deploy a Flow as follows:

```
globus-automate flow deploy --title <title> \
    --definition <Flow definition JSON> --input-schema <Input schema JSON> \
    --visible-to <urn of user or group which can see this Flow> \
    --runnable-by <urn of user or group which can run this Flow> \
    --administered-by <urn of user or group who can maintain this flow>
```

When deployed this way, only the identity that deployed the Flow will be able to view the Flow and only they will be able to run an instance of the Flow. When deploying, it's possible to specify who should be able to see and run the Flow. Using the `visible_to` flag, you can indicate which Globus identities can view the deployed Flow, or set it to `public`, which creates a Flow viewable by anyone. Using the `runnable_by` flag, you can indicate which Globus identities can run an instance of the deployed Flow, or set a value of `all_authenticated_users` which allows any authenticated user to run an instance of the Flow.

Below, we demonstrate how to deploy a Flow that is `visible_to` a single Globus group and `runnable_by` any authenticated user:

```
globus-automate flow deploy --title <title> \
    --definition <Flow definition JSON> \
    --input-schema <Input schema JSON> \
    --visible-to urn:globus:groups:id:00000000-0000-0000-0000-000000000000 \
    --runnable-by all_authenticated_users
```

Once deployed, the output will be the Flow description as displayed by the `flow display` command above. These command line options provide the values for the similarly named fields in the Flow description. Of these, only `title` and `definition` are required. To aid users in using your Flow, we highly recommend the use of `input-schema` as it provides them both a form of documentation and assurance at run-time that the input they provide is correct for executing the Flow. By providing a value or values to `administered-by` you grant rights to others for updating or eventually removing the Flow you have deployed. Commands for updating and removing flows are as follows.

```
globus-automate flow update --title <title> \
    --definition <Flow definition JSON>  --input-schema <Input schema JSON> \
    --visible-to <urn of user or group which can see this Flow> \
    --runnable-by <urn of user or group which can run this Flow> \
    --administered-by <urn of user or group who can maintain this flow> \
    <flow_id>
```

This will update any of the fields or description of the Flow, including the Flow definition itself. Note the `flow_id` field is present at the end of the command line.

Deleting a Flow is done via:

```
globus-automate flow delete <flow_id>
```

Care should be taken when issuing this command. There is no further prompting to ensure the flow should really be deleted. After deletion, no record of the Flow definition or its execution history (i.e. the `flow action-*` commands) is maintained.

The bulk of the effort in creating flows is in authoring their definition which is covered in the section *Authoring Flows for the Globus Flows Service*.

CHAPTER

FOUR

LICENSE

CHAPTER

FIVE

AUTHORING FLOWS FOR THE GLOBUS FLOWS SERVICE

The Globus Flows Service provides users with the ability to easily define compositions of Actions (henceforth referred to as Flows) to perform a single, logical operation. Definition of such Flows requires an easy to read, author, and potentially visualize method of defining the Flows. For this purpose, the Flows service starts from the core of the Amazon States Language. In particular, the general structure of a Flow matches that of a States Language State Machine in particular matching the requirements defined for Top-Level Fields including the properties:

- `States`
- `StartAt`
- `Comment`

Additionally, general concepts from the States Language and its method of managing state for the State Machine/Flow are maintained. Concepts such as Input and Output Processing are handled in the same manner (see note below for an important exception). In particular, paths within the state of the Flow are referenced with a `$.` prefix just as defined in the States Language.

The following state types are supported in Flows in nearly (see note below) the same manor as defined in the States Language:

- Pass
- Choice
- Wait
- Fail

Note: The exception is the use of the `OutputPath` property in the `Pass` or `Choice` states. `OutputPath` is not allowed in a Flow definition. Instead, the `ResultPath` must always be used to specify where the result of a state execution will be stored into the state of the Flow.

Other state types defined in the Amazon States Language are not supported and will be rejected at deployment time. The Flows system adds two new state types `Action` and `ExpressionEval` for invoking Actions and updating the state of a running Flow via an expression language. These are defined below.

5.1 `Action` State type

As Actions are the core building block for most concepts in Globus Automate, Action invocation takes on a central role in the definition of Flows. Actions are invoked from a Flow using the state type `Action`. We describe the structure of an `Action` state via the following example which is described in detail below:

```
{
  "Type": "Action",
  "ActionUrl": "<URL to the Action, as defined above for various Actions>",
  "InputPath": "$.Path.To.Action.Body",
  "Parameters": {
    "constant_val": 10,
    "reference_value.$": "$.Path.To.Value",
    "expression_value.=": "'Constant string ' + Path.To.SuffixString",
    "nested_value": {
      "child_const_val": true,
      "child_ref_val.$": "$.Child.Val.Path"
    },
    "secret_value": "MyPassword",
    "__Private_Parameters": ["secret_value"]
  },
  "ResultPath": "$.ActionOutput",
  "WaitTime": 3600,
  "ExceptionOnActionFailure": true,
  "RunAs": "User",
  "Catch": [
    {
      "ErrorEquals": ["ActionUnableToRun"],
      "Next": "RunFailureHandler"
    },
    {
      "ErrorEquals": ["ActionFailedException"],
      "Next": "ActionFailureHandler"
    }
  ],
  "Next": "FollowingState",
  "ActionScope": "<Scope String for the Action, as defined above for various Actions>",
  "End": true
}
```

The properties on the `Action` state are defined as follows. In some cases, we provide additional discussion of topics raised by specific properties in further sections below this enumeration.

- `Type` (required): As with other States defined by the States Language, the `Type` indicates the type of this state. The value `Action` indicates that this state represents an Action invocation.
- `ActionUrl` (required): The base URL of the Action. As defined by the Action Interface, this URL has methods such as `/run`, `/status`, `/cancel` and so on defined to manage the life-cycle of an Action. The Action Flow state manages the life-cycle of the invoked Action using these methods and assumes that the specific operations are appended to the base URL defined in this property. For Globus operated actions, the base URLs are as defined previously in this document.
- `InputPath` or `Parameters` (mutually exclusive options, at least one is required): Either `InputPath` or `Parameters` can be used to identify or form the input to the Action to be run as passed in the `body` of the call to the action `/run` operation.

  - `Parameters`: The Parameters property is defined as an object that becomes the input to the Action. As such, it becomes relatively plain in the `Action` state definition that the structure of the `Parameters` object matches the structure of the body of the input to the Action being invoked. Some of the fields in the `Parameters` object can be protected from introspection later so that secret or sensitive information, such as credentials, can be encoded in the parameter values without allowing visibility outside the flow, including by those running the Flow. The private parameter functionality is described in *Protecting Action and Flow State*. Values in `Parameters` can be specified in a variety of ways:
    * **Constants**: Simply specify a value which will always be passed for that property. Constants can be any type: numeric, string, boolean or other objects should an action body specify sub-objects as part of their input. When an object is used, each of the properties within the object can also be of any of the types enumerated here.
    * **References**: Copies values from the state of the flow to the name given. The name must end with the sequence `.$` to indicate that a reference is desired, and the string-type value must be a Json Path starting with the characters `$.` indicating the location in the Flow run-time state that values should be retrieved from.
    * **Expressions**: Allow values to be computed as a combination of constants and references to other state in the Flow's run-time. This provides a powerful mechanism for deriving parameter values and is defined more fully below in *Expressions in Parameters*
  - `InputPath`: Specifies a path within the existing state of the Flow here the values to be passed will be present. Thus, use of `InputPath` requires that the proper input be formed in the Flow state.

- `ResultPath`: Is a Reference Path indicating where the output of the Action will be placed in the state of the Flow run-time. The entire output returned from the Action will be returned including the `action_id`, the final `status` of the Action, the `start_time` and `completion_time` and, importantly, the `details` containing the action-specific result values. If `ResultPath` is not explicitly provided, the default value of simply `$`, indicating the root of the Flow state, is assumed and thus the result of the Action will become the entire Flow state following the `Action` state's execution. Typically this is not the desired behavior, so a `ResultPath` should almost always be included.
- `WaitTime` (optional, default value `300`): The maximum amount time to wait for the Action to complete in seconds. Upon execution, the Flow will monitor the execution of the Action for the specified amount of time, and if it does not complete by this time it will abort the Action. See *Action Execution Monitoring* for additional information on this. The default value is `300` or Five Minutes.
- `ExceptionOnActionFailure` (optional, default value `true`): When an Action is executed but is unable complete successfully, it returns a `status` value of `FAILED`. It is commonly useful to treat this "Action Failed" occurrence as an Exception in the execution of the Flow. Setting this property to `true` will cause a Run-time exception of type `ActionFailedException` to be raised which can be managed with a `Catch` statement (as shown in the example). Further details on discussion of the `Catch` property of the Action state and in the *Managing Exceptions* section. If the value is `false`, the status of the Action, including the value of `FAILED` for the status value will be placed into the Flow state as referenced by `ResultPath`.
- `RunAs` (optional, default value `User`): When the Flow executes the Action, it will, by default, execute the Action using the identity of the user invoking the Flow. Thus, from the perspective of the Action, it is the user who invoked the Flow who is also invoking the Action, and thus the Action will make authorization decisions based on the identity of the User invoking the Flow. In some circumstances, it will be beneficial for the Action to be invoked as if from a user identity other than the user who invoked the Flow. See *Identities and Roles, Scopes and Tokens* for additional information and a discussion of use cases for providing different `RunAs` values.
- `Catch`: When Actions end abnormally, an Exception is raised. A `Catch` property defines how the Exception should be handled by identifying the Exception name in the `ErrorEquals` property and identifying a `Next` state to transition to when the Exception occurs. If no `Catch` can handle an exception, the Flow execution will abort on the Exception. A variety of exception types are defined and are enumerated in *Managing Exceptions*.

- `ActionScope` (optional): The scope string to be used when authenticating to access the Action. In most cases, this values is unneeded because the required scope can be determined by querying the Action Provider using the provided `ActionUrl`. If you are using a non-standard compliant Action which does not publish its `scope`, this can be provided to avoid attempting to query the non-compliant Action provider.
- `Next` or `End` (mutually exclusive, one required): These indicate how the Flow should proceed after the Action state. `Next` indicates the name of the following state of the flow, and `End` with a value `true` indicates that the Flow is complete after this state completes.

5.2 Protecting Action and Flow State

At times, portions of a Flow state may need to be secret or protected from the various operations, like status and log, which can be used to monitor and observe the state of a Flow execution. For example, some Actions may require credentials or keys to authenticate or permit access. These items should not be visible to some users, particularly when they are encoded (e.g. in Parameter constants) by the Flow author. There are two areas where these values may be stored or encoded: in `Parameters` to Actions, and within the state of the Flow at run-time. The service provides mechanisms for protecting information in both cases.

For `Parameters`, a list with special property name `__Private_Parameters` may be placed in the `Parameters` object indicating which other Parameters should be protected. These values will be protected in two ways:

- Users that lookup the Flow in the service will not see the `Parameters` which are specified in the `__Private_Parameters` list unless they have the `flow_administrator` or `flow_owner` role on the Flow.
- When the state of a run of the Flow is returned, values for these `Parameters` will not be returned in the status or log of the Flow's execution.

For simplicity, the values in the `__Private_Parameters` list may include the "simple" name even when the parameter name is a Reference or Expression. For example, if a parameter value has the form `"SecretValue.$": "$.Path.To.Secret"` the value in the `__Private_Parameters` list may be simply `SecretValue` omitting the trailing `.$` which identifies the parameter as a reference. Similarly for expression parameters, the trailing `.=` may be omitted. The `__Private_Parameters` list may be applied at any nesting level of the Parameters. Thus, in the following `Parameters` definition:

```
{
  "Parameters": {
    "server_info": {
      "URL": "https://example.com",
      "user_name": "FlowUser",
      "password": "my_password",
      "__Private_Parameters": ["password"]
    }
  }
}
```

The `password` property within the `server_info` object would be omitted from output of any state of the Flow retrieved by any user.

To protect the state of the Flow's run-time, any property which starts with the prefix `_private` will be omitted from Flow introspection. Thus, if protected values need to be stored within the Flow state, they could be stored in a property with a name like `_private_secret_property` or in an object simply having the name `_private` as that object, starting with the prefix will entirely be omitted from the output. As an example, the following flow state would not be visible:

```
{
  "_private": {
      "user_name": "FlowUser",
      "password": "my_password",
  }
}
```

However, the properties *MAY* still be referenced as part of a reference path such as in an Action parameter. Thus, the reference path `$._private.password` could be used and the value `my_password` would be used for the parameter. In such a case, that parameter would also most likely need to appear in the `__Private_Parameters` list to prevent the value from being shown when the state of the particular Action is displayed to a user. Thus, the state protection via `_private` property names and the enumeration of protected parameters via `__Private_Parameters` will often be used in tandem.

5.3 Expressions in Parameters

Action Parameters allow the inputs to an Action to be formed from different parts of the Flow run-time state. However, the reference approach requires that the exact value needed be present in the Flow's state. If the required value is somehow to be derived from multiple values in the Flow state, reference parameters are not sufficient. Thus, we introduce expression type parameters which may evaluate multiple parts of the state to compute a single, required value.

The syntax of an expression paramter takes the following form:

```
{
  "computed_param.=": "<state_val1> <op> <state_val2> <op> ..."
}
```

The syntax for the expression largely follows what is expected in common expression languages. This includes common arithmetic operators on numeric values as well as operations on strings (e.g. string concatenation via a + operation) and on lists (similarly the + operator will concatenate lists).

The values in the state of the flow may be used in the expression and are denoted as `<state_valN>` above. For the following description, assume that the input to (or current state of) a Flow Run is as follows:

```
{
  "foo": "bar",
  "list_val": [1, 2, 3],
  "object_val": {
    "sub_val1": "embedded",
    "sub_val2": "also_embedded"
  }
}
```

The `state_val` values can be specified as the simple names of the properties in the state of the running flow and allows for indexing into lists and into embedded objects similar to Python. Thus, the following would be a valid expression: `foo + ' ' + object_val.sub_val1` which would yield the string `bar embedded`. Note the use of + to mean string concatenation and the dot-separated naming of the field of the object.

Constants may also be used between operators, it is important to remember that within an expression, a string type value must be enclosed in quotes (either single quote characters as above which is often easier because they do not need to be escaped within a JSON string or double quotes).

5.3.1 Using Functions in Expressions

In addition to basic arithmetic operations, a few *functions* may be used. Functions are invoked with the general form: `function_name(param1, param2)`. Thus, an expression may, for example, take the form `val1 + function(param1)`. The functions currently supported are:

- `pathsplit`: This function may be used to break apart a "path" type string value. Paths are a series of path element names separated by `/` characters. The return value from the `pathsplit` function is an array of two elements: the first element is the path prior to the last element. This is also aware of a special "root path" of the form `/~/` as defined by Globus Transfer so that this string will never be "split". Examples:
  - `pathsplit("/foo/bar/blech")` returns `["/foo/bar", "blech"]`
  - `pathsplit("/~/path")` returns `["/~/", "path"]`
- `is_present`: This function checks for the existence of a value in the state of the input Parameters. It is similar to the IsPresent operator in the Amazon States Language. It takes in a reference to a value in the state, *as a string*, and returns `true` if the value exists, and `false` if not. This can be used to insure that a value is present before using it in a further expression such as: `x if is_present('x') else 10` which would use the conditional expression to check for presence of the property `x` and sets a constant if it is not present. This helps to avoid accessing properties that are not defined which would cause an error.
- `getattr`: This function will return a value from the state of the input if it is present, and, optionally, a default value if it is not present. Examples:
  - `getattr('x', 10)`: returns the value of property `x` if it is present, and the constant 10 if not (equivalent to the `is_present` example above.
  - `getattr('missing_property')`: Would return a `null` value if the ``missing_property`` value is not present in the state.

5.4 Identities and Roles, Scopes and Tokens

The `RunAs` property of an `Action` state can be used to control the identity associated with executing the Action. In most cases, it will be appropriate to have the Action invoked as the same identity that invoked the Flow. This is the default behavior, so no value for `RunAs` is needed to get this behavior. However, other scenarios may require a single Flow execution to invoke various Actions using different identities or roles. The `RunAs` property of the `Action` state provides two additional types of roles that can be specified:

- `Flow`: When the value is `Flow`, the Action will be invoked as the identity of the Flow itself. Because every Flow is registered with the Globus Auth system so that it can authenticate requests to be run, it also has a unique identity in Globus Auth. This identity can be used to invoke other Actions. Thus, once the Flow is deployed, the Globus Auth identity of the Flow is known, and can be configured in the authorization state of various Actions for permission. To help with this form of configuration, the information provided by a flow using the command `globus-automate flow list` or `globus-automate flow display` includes two properties which help identity the Flow. The first is `principal_urn` which provides the URN form of the identity for the Flow which is used by many Actions and other Globus services to specify identities. The other property is `globus_auth_username` which is another common method of naming a Globus Auth identity.
- An arbitrary "role name" can also be specified as in `"RunAs": "AdminUser"`. The identity for this role will be determined by an additional Globus Auth access token which is passed into the Flow at run-time as part of the initial state. The flows service will use this token when invoking the Action and so the Action will see the request as if coming from the user associated with this token. We describe how these role-specific tokens are passed next.

Note: When a Flow is run, the identity of the running user is determined by examining the token passed on the header of the HTTP request, and, as described in the next section, other tokens may be passed in the body of the request. In either case, the Flows service will validate the token by interacting with the Globus Auth service. These interactions with Globus Auth require additional time when a Flow is being started. To help alleviate this overhead, the Flows service will retain (cache) results from token validity checks for up to 30 seconds. That is, if the same token is presented more than once within 30 seconds, the results from the previous check will be re-used.

Thus, if a user should request that a token's validity be rescinded, it is _possible_ that use of the token may be considered valid for up to 30 seconds after the time the user rescinds the token's validity.

5.4.1 Providing Role-Specific Tokens

When `RunAs` specifies a role name, corresponding tokens must be generated and provided to the Flow at run-time. The necessary information to generate any Globus Auth token is the name of the scope to which the token should be generated. So that generated tokens are as specific as possible, the Flows service creates a separate scope for each role which appears as part of a `RunAs` property. These scope strings are present in the Flow description under the property `globus_auth_scopes_by_RunAs`. This will be a JSON object with the property names matching the roles named in `RunAs` and the values being the Globus Auth scope string. For example, if roles named `Admin` and `Curator` were present in the Flow definition, the Flow description would contain an object like:

```
{
  "Admin": "<Globus Auth Scope String for Admin>",
  "Curator": "Globus Auth Scope String for Curator>"
}
```

When invoking the Flow (e.g. via `globus-automate flow run`) the flow input would be required to contain the access tokens for each of the roles in a similar JSON object called `_tokens` as follows:

```
{
  "_tokens": {
    "Admin": "<Globus Auth access token for Admin>",
    "Curator": "Globus Auth access token for Curator>"
  }
}
```

Note: If the author of a Flow provides an `input_schema` for their Flow, the schema should specify that the `_tokens` property should be present with this structure. Otherwise, the Flows service will reject the input prior to running the Flow.

The method for generating the required tokens is outside the scope of this document. The approach will use of the Globus Auth API and typically the Globus SDK. In particular, the section on obtaining tokens is a good starting point.

5.5 Action Execution Monitoring

`Action` states will block until the executed action reaches a completion state with status value either `SUCCEEDED` or `FAILED` or when the `WaitTime` duration is reached. Within this time, the Flow will periodically poll the Action to determine if it has reached a completion state. The interval between polls doubles after each poll ("exponential back-off") up to a maximum interval between polls of 10 minutes. Thus, detection of the completion will not be instantaneous compared to when the action "actually" completes and may be delayed up to the maximum poll interval of 10 minutes.

It is important to remember that this delay between an Action's actual completion and it being detected by the Flow service can occur. A user running a flow may observe or receive another form of notification (such as an email from Globus Transfer) that an Action has completed prior to the Flows service polling to discover the same progress has occurred. This is an inherent property of the system.

5.6 Managing Exceptions

Failures of Action states in the Flow are exposed via Exceptions which, as described above, can be handled via a `Catch` property on the Action state. The form of the `Catch` is as shown in the example, but the types of exceptions need to be discussed in more detail. There are three forms of exceptions that impact an Action execution:

- `ActionUnableToRun`: This exception indicates that the initial attempt to run the Action failed and no action whatsoever was initiated. The output of the exception contains the error structure returned by the Action. This condition will always result in an exception.
- `ActionFailedException`: This indicates that the Action was able initiated but during execution the Action was considered to have failed by the return of an Action status with the value `FAILED`. This exception will only be raised if the property `ExceptionOnActionFailure` is set to true. This allows the Action failure to be handled by checking the result or by causing an exception. Either approach is valid and different users and different use cases may lend themselves to either approach. In either case, the output will contain the same Action status structure a completed action will contain, but the `status` value will necessarily be `FAILED`.
- `ActionTimeout`: When the `WaitTime` for an `Action` state is exceeded, this exception will be raised. The status of the most recent poll of the `Action` will be contained within the body of the exception.

5.7 Pre-Populated Run-time State

Basic information about the flow's state and the user invoking the Flow is provided through a "virtual", read-only property available at the JSONPath `$._context`. This path may be used in a path for a `Parameters` value on an Action or Pass state type, or in expressions which are evaluated when generating `Parameters` values as described above. This allows the Flow to use these values as necessary for passing into Actions as parameters. As this is a read-only value, the `_context` cannot be overwritten by using the path in a `ResultPath` on any state. The `_context` value is itself an object containing the following properties:

Property name	Description
flow_id	The id of the deployed Flow that is executing
run_id	The unique id assigned to **this execution** of the Flow
username	The Globus Auth username for the user invoking the Flow
email	The email address for the user invoking the Flow
user_id	The Globus Auth user id for the user invoking the Flow (in URN format)
identities	A list of all identities associated with the user invoking the Flow (in URN format)
token_info	A child object containing the fields exp, iat, and nbf (described below)

The `token_info` fields are defined as follow:

- `exp`: Timestamp, measured in the number of seconds since January 1 1970 UTC, indicating when this token will expire.
- `iat`: Timestamp, measured in the number of seconds since January 1 1970 UTC, indicating when this token was originally issued.
- `nbf`: Timestamp, measured in the number of seconds since January 1 1970 UTC, indicating when this token is not to be used before.

5.8 `ExpressionEval` State type

The `Action` state type provides a method of evaluating expressions to create Parameter values for passing to the action, and the `Pass` state, defined in the States Language, provides a means of moving or re-arranging the Flow's run-time state by specifying input Parameters and new locations via the `ResultPath`. In some cases, the combination of the two capabilities is desired: the ability to compute results for Parameters as in the `Action` state and the simple storage of the new values, as in the `Pass` state. This is the role of the `ExpressionEval` state type. It can be thought of as an `Action` without the Action invocation, or a `Pass` where `Parameters` may contain expressions.

A primary situation in which this state type will be used is when determining a value to be tested in a `Choice` state type. The `Choice` state type can only read single values from the run-time state of the Flow, so if, for example, a value on which a `Choice` condition needs to be applied must be combined from separate parts of the Flow run-time state. The computed value can then be referenced in the `Variable` property of the Choice. Another use is to compute a "final" for the Flow to be stored in the state of the Flow and therefore seen in the output of the Flow upon completion.

An example structure for an `ExpressionEval` state is as follows:

```
{
  "Type": "ExpressionEval",
  "Parameters": {
    "constant_val": 10,
    "reference_value.$": "$.Path.To.Value",
    "expression_value.=": "'Constant string ' + `$.Path.To.SuffixString`",
    "nested_value": {
      "child_const_val": true,
      "child_ref_val.$": "$.Child.Val.Path"
    },
    "secret_value": "MyPassword",
    "__Private_Parameters": ["secret_value"]
  },
  "ResultPath": "$.final_result",
  "End": true
}
```

All of the properties of the `ExpressionEval` state have the same meaning as described in the `Action` state. The `ExpressionEval` state cannot use the `InputPath` property (`Pass` is appropriate if moving state from an `InputPath` to a `ResultPath` is needed), so `Parameters` must always be present. Just like in `Action` the `Parameters` may have constant, reference or expression types and portions of the state can be protected using a `__Private_Parameters` list. Like `Action`, this state must have either a `Next` or an `End: true`.

5.9 Globus Web App Custom Formats

The Globus web app supports a JSON schema format in order to make starting flows a little more user friendly on the webapp.

5.9.1 globus-collection

`globus-collection` as a `format` in your `input_schema` will signal to the webapp to show a custom input field for searching for and selecting a Globus collection on the Guided tab when starting a Flow.

This example input schema shows how to configure a basic Transfer flow using this format:

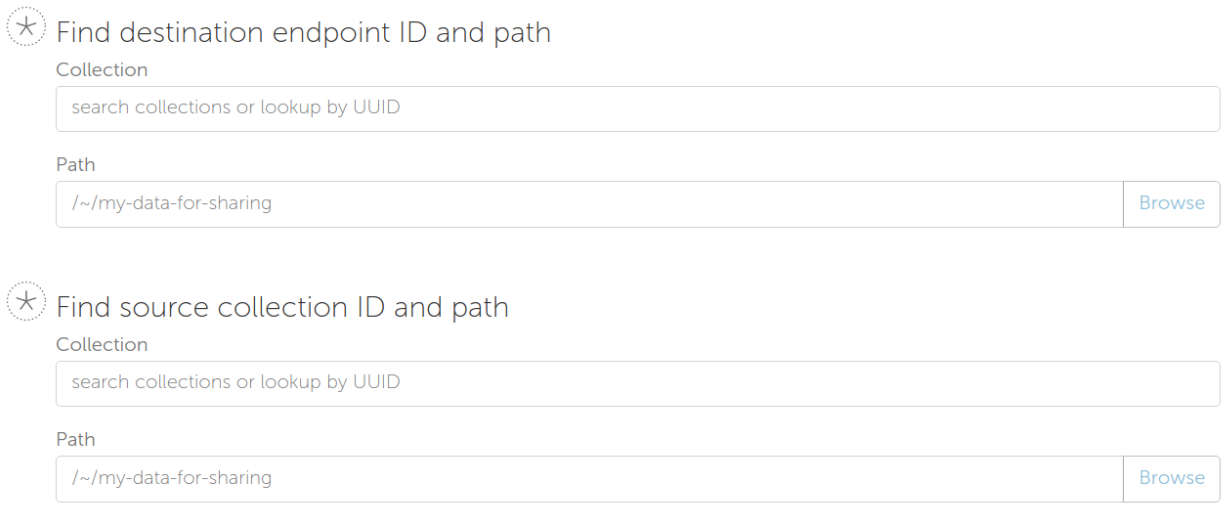
```
{
    "additionalProperties": false,
    "properties": {
        "source": {
            "type": "object",
            "format": "globus-collection",
            "title": "Find source collection ID and path",
            "required": [
                "id",
                "path"
            ],
            "properties": {
                "id": {
                    "type": "string",
                    "format": "uuid"
                },
                "path": {
                    "type": "string"
                }
            },
            "additionalProperties": false
        },
        "destination": {
            "type": "object",
            "format": "globus-collection",
            "title": "Find destination endpoint ID and path",
            "required": [
                "id",
                "path"
            ],
            "properties": {
                "id": {
                    "type": "string",
                    "format": "uuid"
                },
                "path": {
                    "type": "string"
                }
            },
            "additionalProperties": false
```

(continues on next page)

(continued from previous page)

```
        },
        "recursive": {
            "type": "boolean",
            "title": "Recursive transfer",
            "description": "Whether or not to transfer a directory recursively, must be␣
↪true when transferring a directory."
        }
    },
    "required": ["source", "destination", "recursive"]
}
```

The above will cause the Globus we application to display a set of inputs that map to the id and path fields for source and destination:



5.10 Important notes about the `globus-collection` format:

- The properties inside `globus-collection` must be named "id" and "path"
- **The `required` field inside `globus-collection` determines behavior of the component as follows:**
  - If both "id" and "path" are required, the component will display and require both collection and path inputs
  - If only "id" is required, the component will only display the Collection input
  - If only "path" is required, the component will display both inputs but only the path field will be required. The collection input is provided to allow browsing of that collection's directory listing
- This format is used to provide a UI component for Globus web app and will not substantively affect Flow usage from the Automate CLI or programmatic access

5.11 Example Flows

5.11.1 "Move (Globus Example)"

Flow ID: `9123c20b-61e0-46e8-9469-92c999b6b8f2`.

A Flow which performs a 'move' operation on a directory by first transferring from a source to a destination and then deleting the directory from the source. The entire directory's contents, including files and subdirectories, will be moved to the destination and then removed from the source.

Note that this flow requires at least one of the collections to be managed under a Globus subscription.

View the Move flow definition in the Globus web app. (You may need to log in first.)

Listing 1: Example Input

```
{
    "source": {
      "id": "ddb59aef-6d04-11e5-ba46-22000b92c6ec",
      "path": "/~/source-directory"
    },
    "destination": {
      "id": "ddb59af0-6d04-11e5-ba46-22000b92c6ec",
      "path": "/~/destination-directory"
    }
    "transfer_label": "Transfer for Generic Move from Globus Tutorial Endpoint 1 to␣
↪Globus Tutorial Endpoint 2",
    "delete_label": "Delete after Transfer for Generic Move from Globus Tutorial␣
↪Endpoint 1 to Globus Tutorial Endpoint 2"
}
```

(Choose different `source.path` and `destination.path` as needed to run this example flow.)

5.11.2 "2 Stage Transfer (Globus Example)"

Flow ID: `79a4653f-f8da-43b6-a581-5d3b345ad575`.

Transfer from source to destination with an intermediate endpoint in-between. Remove from intermediate after completion.

Note that this flow requires at least one of the collections to be managed under a Globus subscription.

View the 2 Stage Transfer flow definition in the Globus web app.

Listing 2: Example Input

```
{
    "source": {
      "id": "ddb59aef-6d04-11e5-ba46-22000b92c6ec",
      "path": "/~/ep1-example-directory/"
    },
    "intermediate": {
      "id": "ddb59af0-6d04-11e5-ba46-22000b92c6ec",
      "path": "/~/ep2-intermediate-directory/"
    },
```

(continues on next page)

(continued from previous page)

```
    "destination__": {
      "id": "ddb59aef-6d04-11e5-ba46-22000b92c6ec",
      "path": "/~/ep1-duplicate-example-directory/"
    }
    "transfer1_label": "This value will be used as a label for the Globus Transfer Task␣
↪to copy data from the source collection to the intermediate collection",
    "transfer2_label": "This value will be used as a label for the Globus Transfer Task␣
↪to copy data from the intermediate collection to the destination collection"
}
```

5.11.3 "Transfer Set Permissions (Globus Example)"

Flow ID: `cdcd6d1a-b1c3-4e0b-8d4c-f205c16bf80c`.

A Flow which performs a Transfer on a directory, gives a user READ permissions on the destination directory and notifies the user of their new data via email. The user running the Flow must have administration privileges over the destination endpoint.

View the Transfer Set Permissions flow definition in the Globus web app.

Listing 3: Example Input

```
{
    "source_endpoint_id": "ddb59af0-6d04-11e5-ba46-22000b92c6ec",
    "source_path": "/share/godata",
    "destination_endpoint_id": "ddb59aef-6d04-11e5-ba46-22000b92c6ec",
    "destination_path": "/~/my-godata",
    "transfer_label": "Transfer for Transfer Set Permissions Flow",
    "user_id": "06a24bef-940e-418a-97bc-48229c64cc99",
    "user_email": "uriel@globus.org"
}
```

(Of course, adjust `user_email` as necessary if you want to test this flow.)

CHAPTER

SIX

RUNNING A FLOW AUTOMATICALLY

It is frequently desirable to run a flow when new data becomes available. This document provides examples for how to accomplish this using watchdog to monitor for filesystem events and using the Globus Automate CLI to run already-defined flows.

6.1 Shell scripting

watchdog provides a CLI tool named `watchmedo`. If you're comfortable with shell scripting then you can use the following command to run a script named `runner.sh`.

Listing 1: `watchmedo.sh` [download]

```
watchmedo shell-command \
    --command 'bash runner.sh "${watch_event_type}" "${watch_src_path}"' \
    --recursive \
    .
```

The `watchmedo` command currently has no way to filter filesystem events. On Linux, `runner.sh` will be run when:

- a file is created (but has not had data written yet)
- data is written to the file (but the file has not been closed yet)
- the file is closed

This will likely result in your flow running far more often than expected. To avoid this, `runner.sh` must filter the incoming filesystem events. Here is an example script that will run a flow with custom input when the filesystem event type is `"closed"`:

Listing 2: `runner.sh` [download]

```
set -eu

# Only run a flow when a new file is created.
if [ "$1" != "closed" ]; then
    exit
fi

FLOW_ID="your-flow-id-here"
FLOW_INPUT=$(cat << EOF
{
    "event": "$1",
```

(continues on next page)

(continued from previous page)

```
    "filename": "$2"
}
EOF
)

globus-automate flow run \
    --label "File change: $1" \
    --flow-input "${FLOW_INPUT}" \
    "${FLOW_ID}"
```

Note: Filesystem events are less granular on Windows platforms. Notably, there is no `"closed"` event type to signal all data have been written to a file. There is only a `"modified"` event type, which may fire multiple times if large files are written and flushed to disk in multiple chunks.

Users on Windows may want to use the Python script below.

6.2 Python scripting

If you know that files will be created or modified in large batches, you may need to write a script to monitor for filesystem events and wait some amount of time for filesystem events to taper off. One way to do this is using the watchdog package.

The script below will monitor for filesystem events of all types. It will only run a flow 60 seconds after the most recent filesystem event is encountered.

Listing 3: `runner.py` [`download`]

```
# The flow to run.
FLOW_ID = "your-flow-id-here"

# The flow will be run X seconds after the most recent filesystem event is received.
# If no filesystem events are ever received, the flow will not be run.
COOL_OFF_TIME_S = 60


import logging
import os
import pathlib
import queue
import sys

logging.basicConfig(
    level=logging.WARNING,  # Eliminate INFO messages from the Globus SDK.
    format="%(asctime)s - %(message)s",
    datefmt="%Y-%m-%d %H:%M:%S",
)
log = logging.getLogger(__name__)

try:
    from globus_automate_client import create_flows_client
```

(continues on next page)

(continued from previous page)

```
except ImportError:
    log.error(
        "The globus_automate_client package is not installed."
        " (Do you need to activate a virtual environment?)"
    )
    sys.exit(1)

try:
    import watchdog
    import watchdog.events
    import watchdog.observers
except ImportError:
    log.error(
        "The watchdog package is not installed."
        " (Do you need to activate a virtual environment?)"
    )
    sys.exit(1)


class Handler(watchdog.events.FileSystemEventHandler):
    def __init__(self, events: queue.Queue):
        self.events = events

    def dispatch(self, event):
        """Put all filesystem events in a queue."""

        self.events.put(event)


def main():
    try:
        path = pathlib.Path(sys.argv[1]).absolute()
    except IndexError:
        path = pathlib.Path(os.getcwd()).absolute()
    log.warning(f"Monitoring {path}")
    log.warning("Press CTRL-C to exit (on Windows, press CTRL-BREAK)")

    event_queue = queue.Queue(maxsize=-1)
    handler = Handler(event_queue)
    observer = watchdog.observers.Observer()
    observer.schedule(handler, str(path), recursive=True)
    observer.start()

    flows_client = create_flows_client()

    try:
        timeout = None
        files = set()
        while True:
            try:
                event = event_queue.get(block=True, timeout=timeout)
            except queue.Empty:
```

(continues on next page)

(continued from previous page)

```
                # .get() timed out.
                # It's now been COOL_OFF_TIME_S seconds since the last filesystem event.
                # Reset the timeout for the next batch of files and run the flow.
                timeout = None
                log.warning(f"Running the flow ({len(files)} paths were modified)")
                flows_client.run_flow(
                    flow_id=FLOW_ID,
                    flow_scope=None,
                    flow_input={
                        "count": len(files),
                    },
                    label=f"[AUTO] File system changes detected ({len(files)} paths)",
                )
                files = set()
            else:
                # .get() returned a filesystem event.
                # Make sure the next .get() call times out after COOL_OFF_TIME_S.
                timeout = COOL_OFF_TIME_S
                files.add(event.src_path)
                event_queue.task_done()
    except KeyboardInterrupt:
        pass
    finally:
        observer.stop()
        observer.join()


if __name__ == "__main__":
    main()
```

CHAPTER

SEVEN

GLOBUS ACTION PROVIDERS

Globus provides and operates a number of `Action Providers` which may be invoked directly or used within Flows. Below is a brief summary of the `Action Providers` being operated including specific information on their URLs, scopes, and a summary of their functionality. Specific input specifications are not provided as they may be retrieved from the `Action Provider` directly via introspection:

```
globus-automate action introspect --action-url <action_url>
```

Or simply click on the URL to view the introspection results in a browser.

Note: When running Globus operated Action Providers the `action` subcommands do not require the use of the `--action-scope` option as these Action Providers are publicly visible. If interacting with a non-publicly visible Provider, all `action` subcommands will require the `--action-scope` option followed with the Action Provider's corresponding scope value.

7.1 List of Globus Operated Action Providers

7.1.1 HelloWorld

URL: https://actions.globus.org/hello_world

Scope: `https://auth.globus.org/scopes/actions.globus.org/hello_world`

Synchronous / Asynchronous: Either

The HelloWorld Action Provider is very simple and is primarily intended for testing and bootstrapping purposes. It can operate in either synchronous or asynchronous modes. In the synchronous mode, only a single string is sent in the body and the response will return containing a constant value (`"Hello": "World"`) and a reply to the input (`"hello": "<input string>"`). To get asynchronous operation, the value `sleep_time` should be included in the input with a value of a number of seconds the Action should take to complete. Subsequent invocations of `/status` will return the state `ACTIVE` until the number of seconds indicated in `sleep_time` have elapsed at which point the status will become `SUCCEEDED`.

Listing 1: Example Input

```
{
    "echo_string": "Hello Globus Automate!",
    "sleep_time": 5
}
```

Listing 2: Try It

```
globus-automate action run --watch \
    --action-url https://actions.globus.org/hello_world \
    --body '{"echo_string": "Hello Globus Automate!","sleep_time": 5}'
```

7.1.2 Globus Transfer - Transfer Data

URL: https://actions.globus.org/transfer/transfer

Scope: `https://auth.globus.org/scopes/actions.globus.org/transfer/transfer`

Synchronous / Asynchronous: Either

The Action Provider "transfer/transfer" uses the Globus Transfer Task API to perform a transfer of data from one Globus Collection to another. The input includes both the source and destination collection ids and file paths within the collection where the source file or folder is located and the destination folder where the transfer should be placed. It also supports indicating that transfers should be performed recursively to traverse the entire source file system tree and allows labeling the transfer should it be viewed directly in the Globus WebApp or via the Globus API or CLI. The body of the action status directly reflects the information returned when monitoring the transfer task using the Globus Transfer API.

Listing 3: Example Input

```
{
    "source_endpoint_id": "go#ep1",
    "destination_endpoint_id": "go#ep2",
    "transfer_items": [
        {
            "source_path": "/~/campus_source/dataset1/",
            "destination_path": "/~/dmz_temp/",
            "recursive": true
        }
    ],
    "notify_on_succeeded": false,
    "notify_on_failed": true,
    "notify_on_inactive": true
}
```

7.1.3 Globus Transfer - Delete Data

URL: https://actions.globus.org/transfer/delete

Scope: `https://auth.globus.org/scopes/actions.globus.org/transfer/delete`

Synchronous / Asynchronous: Asynchronous

Globus Transfer / delete data works much like the "Transfer / transfer" provider. It takes a source collection and path as inputs and uses the Globus Transfer Task API to initiate an asynchronous delete operation. Also like the transfer operation, labels and recursive options may be set. The status body comes directly from the Task status in the Globus Transfer API.

7.1.4 Globus Transfer - Set/Manage Permissions

URL: https://actions.globus.org/transfer/set_permission

Scope: `https://auth.globus.org/scopes/actions.globus.org/transfer/set_permission`

Synchronous / Asynchronous: Synchronous

The set permission Action Provider uses the Globus Transfer ACL API to set or manage permissions on a folder or file. The body of the request indicates whether the permission rule is to be created, updated or deleted using the `operation` property. For update or delete, the `rule_id` of a previously created permission rule must also be provided.

As the Globus Transfer API returns a status directly (rather than a task identifier), the Action Provider behaves in a synchronous manner, returning the Transfer API result.

Listing 4: Example Input to Set Permissions

```
{
  "operation": "CREATE",
  "endpoint_id": "ddb59aef-6d04-11e5-ba46-22000b92c6ec",
  "path": "/~/shared/",
  "principal_type": "all_authenticated_users",
  "principal": "",
  "permissions": "r"
}
```

Listing 5: Example Input to Delete Permissions

```
{
  "operation": "DELETE",
  "endpoint_id": "ddb59aef-6d04-11e5-ba46-22000b92c6ec",
  "rule_id": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee"
}
```

Listing 6: Example Input to Update Permissions

```
{
  "operation": "UPDATE",
  "endpoint_id": "ddb59aef-6d04-11e5-ba46-22000b92c6ec",
  "rule_id": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
  "permissions": "rw"
}
```

7.1.5 Globus Transfer - List Directory Contents

URL: https://actions.globus.org/transfer/ls

Scope: `https://auth.globus.org/scopes/5fac2e64-c734-4e6b-90ea-ff12ddbf9653/transfer_ls`

Synchronous / Asynchronous: Synchronous

The Globus Transfer ls Action Provider uses the Globus Transfer Directory API to retrieve a listing of contents from an (endpoint, path) pair. Although providing a path is optional, the default path used depends on endpoint type and it is best to explicitly set a path. This Action Provider supports all options as defined in the List Directory Contents Transfer API documentation.

Additionally, it adds two features not found in the base Globus Transfer List Directory operation.

1. An input Boolean parameter `path_only` may be specified which indicates that regardless of the file type of the path, we only will return information about the path itself. This changes the behavior when the path references a folder. Rather than returning the *contents* of the folder, this will return information about the folder itself. Thus, when this option is present, the output will contain only one file entry.
2. For each file entry returned, an additional property, `is_folder`, is included. This is set to `true` if the file type is folder. This can be helpful, particularly within Flows, to perform branching or other logic dependent on whether a path points to a folder or a regular file.

7.1.6 Globus Transfer - Make Directory

URL: https://actions.globus.org/transfer/mkdir

Scope: `https://auth.globus.org/scopes/5fac2e64-c734-4e6b-90ea-ff12ddbf9653/transfer_mkdir`

Synchronous / Asynchronous: Synchronous

The Globus Transfer mkdir Action Provider uses the Globus Transfer Make Directory API to create a new directory on an endpoint. The input is simply the id for endpoint where the directory will be created and the path on the endpoint to the directory to be created.

Listing 7: Example Input to Make Directory

```
{
  "endpoint_id": "ddb59aef-6d04-11e5-ba46-22000b92c6ec",
  "path": "/~/new_folder"
}
```

7.1.7 Globus Transfer - Get Collection Information

URL: https://actions.globus.org/transfer/collection_info

Scope: `https://auth.globus.org/scopes/5fac2e64-c734-4e6b-90ea-ff12ddbf9653/transfer_collection_info`

Synchronous / Asynchronous: Synchronous

The Globus Transfer ls Action Provider uses the Globus Transfer Get Collection API to get information about a Globus Collection. The information returned is the same as defined by the Globus Transfer API with one addition: a property `is_managed` will be set to `true` if there is a `subscription_id` associated with the collection, and `false` if not. This allows, for example, branching within a Flow (using a `Choice` state type) based on whether a collection/endpoint is managed under a Globus subscription.

7.1.8 Globus Search - Ingest

URL: https://actions.globus.org/search/ingest

Scope: `https://auth.globus.org/scopes/actions.globus.org/search/ingest`

Synchronous / Asynchronous: Asynchronous

Records may be added to an existing Globus Search index using the Search / ingest Action Provider. The input to the Action Provider includes the id of the Search index to be added to and the data, in the Search-defined `GMetaEntry` format. The user calling the Action Provider must have permission to write to the index referenced. Globus Search will process the ingest operation asynchronously, so this Action Provider also behaves in an asynchronous fashion: requests to update the state of an Action will reflect the result from updating the state of the ingest task in Globus Search. Since

Globus Search does not support cancellation of tasks, this Action Provider also does not support cancellation of its Actions.

Listing 8: Example Input

```
{
    "search_index": "6077d057-989d-41e2-b60b-91fe001f0687",
    "subject": "http://example.com/foo",
    "visible_to": [
        "public"
    ],
    "content": {
        "testing": {
            "hello": "world"
        }
    }
}
```

7.1.9 Globus Search - Delete

URL: https://actions.globus.org/search/delete

Scope: `https://auth.globus.org/scopes/5fac2e64-c734-4e6b-90ea-ff12ddbf9653/search_delete`

Synchronous / Asynchronous: Asynchronous

Subjects or entries may be removed from an existing Globus Search index using the Search Delete Action Provider. The input to the Action Provider includes the Search index id to delete the data from. The body also specifies the type of delete operation to execute via the `delete_by` parameter. It's value may be `entry` to delete a single entry, `subject` to delete a subject, or `query` to remove data matching a Search query.

The user calling the Action Provider must have permission to write to the index referenced. Globus Search will process the delete operation asynchronously, so this Action Provider also behaves in an asynchronous fashion. Requests for the state of an Action will lookup the state of the `task_id` in Globus Search, ensuring the status remains up-to-date. Since Globus Search does not support cancellation of tasks, this Action Provider also does not support cancellation of its Actions.

Listing 9: Example Input to Delete by Subject

```
{
    "delete_by": "subject",
    "search_index": "00000000-0000-0000-0000-000000000000",
    "subject": "http://example.com/foo"
}
```

Listing 10: Example Input to Delete by Query

```
{
    "delete_by": "query",
    "search_index": "00000000-0000-0000-0000-000000000000",
    "q": "a search with filtering and faceting",
    "filters": [
        {
            "type": "range",
```

(continues on next page)

(continued from previous page)

```
            "field_name": "path.to.date",
            "values": [
                {
                    "from": "*",
                    "to": "2014-11-07"
                }
            ]
        }
    ],
    "query_template": "some_query_template"
}
```

7.1.10 Send Notification - Email

URL: https://actions.globus.org/notification/notify

Scope: `https://auth.globus.org/scopes/5fac2e64-c734-4e6b-90ea-ff12ddbf9653/notification_notify`

Synchronous / Asynchronous: Synchronous

The Send notification / email Action Provider presently supports sending of email messages to a set of email addresses. The request to send the email contains the standard components of an email: sender, receiver(s), subject and body. The mimetype of the body may be specified so that either HTML or text formatted messages may be sent. The body also supports the notion of variable substitution or "templating." Values in the body may be specified with a dollar sign prefix ($), and when values are provided in the `body_variables` property of the request, the template value will be substituted with the corresponding value from the `body_variables`.

The other important component of the request to this action provider is the email sending credentials. Credentials are provided to allow the provider to communicate with the service used for sending the email. Presently, two modes of sending email are supported: SMTP and AWS SES. When SMTP is provided, the username, password and server hostname are required. When AWS SES is provided, the AWS access key, AWS access key secret and the AWS region must be provided. As this service is synchronous and stateless, the requester can be assured that these credentials will not be stored. The Action Provider will return success as long as the email service accepts the message. It cannot guarantee successful delivery of the message including an inability to deliver the message due to an improper recipient address.

Listing 11: Example Input with Complex Templating

```
{
    "notification_method": "email",
    "notification_priority": "high",
    "body_template": "<html><body><h1>Hello $Name</h1></body></html>",
    "body_mimetype": "text/html",
    "body_variables": {
        "Name": "Dude"
    },
    "destination": "user@globus.org",
    "sender": "user@globus.org",
    "subject": "Testing From Automate CLI",
    "send_credentials": [
        {
            "credential_method": "email",
```

(continues on next page)

(continued from previous page)

```
            "credential_type": "ses",
            "credential_value": {
                "aws_access_key_id": "SECRET",
                "aws_secret_access_key": "SECRET",
                "region_name": "us-east-1"
            }
        }
    ]
}
```

Listing 12: Example Input with Simple Formatting

```
{
    "body_mimetype": "text/html",
    "body_template": "Hello there",
    "destination": "DESTINATION@example.org",
    "notification_method": "any",
    "notification_priority": "high",
    "sender": "SENDER@example.org",
    "subject": "Testing Notification",
    "send_credentials": [
        {
            "credential_type": "ses",
            "credential_value": {
                "aws_access_key_id": "AWS_ACCESS_KEY_ID",
                "aws_secret_access_key": "AWS_SECRET_ACCESS_KEY",
                "region_name": "us-east-1"
            }
        }
    ]
}
```

7.1.11 Wait for User Option Selection

URL: https://actions.globus.org/weboption/wait_for_option

Scope: `https://auth.globus.org/scopes/5fac2e64-c734-4e6b-90ea-ff12ddbf9653/weboption_wait_for_option`

Synchronous / Asynchronous: Asynchronous

Flows or other clients which desire to provide users a method of selecting an option from a fixed set may use the Wait for User Option Selection Action Provider. The Action Provider can operate in one of two modes.

In the first mode, a list of options are created which are automatically selected by any access to a corresponding URLs. For each option, a name, a URL suffix, and a message or text which is returned in the HTTP response of the selection operation is provided. The URL suffix is registered with the Action Provider and is monitored at the URL `https://actions.globus.org/weboption/option/<url_suffix>`. Any HTTP access to the URL is considered a selection of that option among all the options defined by the input to the Action and the Action will transition to a `SUCCEEDED` status. Each of the options may be protected for access only via specific Globus identities by setting values on the `selectable_by` list. A direct HTTP access may present a Bearer token for authorization using the same scope as used for accessing the other operations on the Action Provider. If no access token is presented, the user will be re-directed to start an OAuth Flow using Globus Auth to authenticate access to the option URL.

In the second mode, in addition to monitoring the provided URL suffixes, a landing page may be hosted which will present the options to a user on a simple web page. The web page may be "skinned" with options for banner text, color scheme and icon as well as introductory text presented above the options. The options are specified in the same manner as in the first mode, but the page presents links which ease selection of those options for end-users. The landing page is also given a URL suffix, and the selection page will be present at `https://actions.globus.org/weboption/landing_page/<url_suffix>`. Selection of an option within the landing page behaves the same as direct selection of an option via its URL as described above. Similar to individual options, the landing page can be protected by setting a `selectable_by` list. As the landing page is intended for use via a browser, it will always start a OAuth Flow to authenticate the user. If `selectable_by` is set on the landing page but not on any of the individual options, the options inherit the same `selectable_by` value defined on the landing page for that Action.

In either mode, once an option has been selected, none of the url suffixes, nor the landing page if configured, in the initial request, will be responded to by the Action Provider: they will return the HTTP not found (error) status 404. Upon completion, the body of the status will include the name and the url suffix for the selected option. The body may also include input on the HTTP data passed when the option's URL was accessed including the query parameters and the body. To include those in the status, flags are set on the definition of the option.

Listing 13: Example Input for Creating Options

```
{
    "options": [
        {
            "name": "b",
            "url_suffix": "option_b",
            "completed_message": "Thank you for selecting 'b'"
        },
        {
            "name": "default",
            "url_suffix": "option_a",
            "completed_message": "Thank you for selecting the default option"
        }
    ]
}
```

Listing 14: Example Input for Creating Options on a Landing Page

```
{
    "landing_page": {
        "url_suffix": "landing_page",
        "header_background": "darkred",
        "header_icon_link": "http://example.com",
        "header_text": "Hey, Make a choice",
        "page_title": "Look at my title",
        "preamble_text": "Please make a very careful decision"
    },
    "options": [
        {
            "name": "b",
            "description": "This is option b",
            "url_suffix": "option_b_new",
            "completed_message": "Thank you for selecting 'b'"
        },
        {
            "name": "default",
```

(continues on next page)

(continued from previous page)

```
            "description": "This is the default option",
            "url_suffix": "option_a_new",
            "completed_message": "Thank you for selecting the default option"
        }
    ]
}
```

Listing 15: Example Input for Creating Options on a Landing Page Limiting Selection

```
{
    "landing_page": {
        "url_suffix": "landing_page",
        "header_background": "darkred",
        "header_icon_link": "http://example.com",
        "header_text": "Hey, Make a choice",
        "page_title": "Look at my title",
        "preamble_text": "A summary of the user's input...",
        "include_text_input_form": true,
        "text_input_form_prompt": "Please provide your reason",
        "display_options_as": "link",
        "selectable_by": [
            "urn:globus:auth:identity:c5ab1c3d-b812-4ef9-acb6-4d84c58db4de"
        ]
    },
    "options": [
        {
            "name": "Yes",
            "description": "Allow the submission",
            "url_suffix": "yes",
            "completed_message": "The submission will be allowed"
        },
        {
            "name": "No",
            "description": "Reject the submission",
            "url_suffix": "no",
            "completed_message": "The submission will be rejected"
        },
        {
            "name": "Maybe",
            "description": "Request the user to re-submit",
            "url_suffix": "maybe",
            "completed_message": "The user will be required to update their submission"
        }
    ]
}
```

7.1.12 Simple Expression Evaluation

Note: Expression Evaluation has been integrated with Action definitions directly (see section *Expressions in Parameters*). Thus, for most use cases, the Simple Expression Evaluation Action Provider described here is not needed and expressions defined on Action definitions within a Flow are preferred.

URL: https://actions.globus.org/expression_eval

Scope: `https://auth.globus.org/scopes/5fac2e64-c734-4e6b-90ea-ff12ddbf9653/expression`

Synchronous / Asynchronous: Synchronous

Evaluation of simple expressions is supported using the simpleeval library and therefore syntax. A single invocation of the Action Provider may evaluate a single expression or multiple expressions. An Expression request consists of up to three parts:

- An `expression` (required) which is a basic "arithmetic" type expression. This *does* include string type operations so an expression like "foo" + "bar" is permitted and performs string concatenation as is common in many programming and scripting languages.
- A set of `arguments` (optional) in a JSON object format. These arguments may be referenced in an expression. So, if there's an expression such as "x + 1" and the arguments contain `{"x": 2}` the result will be `3`.
- A `result_path` (optional) which is a path where the result will be stored. It may be in "Reference Path" format as defined in the AWS Step Functions State Machine Language specification or it may simply be a dot separated string of the path elements. In either case, the path indicates where in the `details` of the returned action status the value for the evaluated expression should be placed. If `result_path` is not present, the result will be stored in the `details` under the key `result`.

A single request may specify multiple expressions to be evaluated by providing an array named `expressions` as in `{"expressions": [{ expression1 }, {expression2}, ...]}` where each of the expressions `expression1` and `expression2` contains the three fields defined for an expression. These will be evaluated in order, and expressions using the same `result_path` will result in previous results being over-written.

7.1.13 Datacite DOI Minting

URL: https://actions.globus.org/datacite/mint/basic_auth

Scope: `https://auth.globus.org/scopes/5fac2e64-c734-4e6b-90ea-ff12ddbf9653/datacite_mint_basic_auth_action_all`

Synchronous / Asynchronous: Synchronous

The Datacite DOI Minting action provider uses the Datacite JSON API to mint DOIs. The main part of the body input is as specified in that API. The additional fields provide the username and password (the "Basic Auth" credentials which is part of the name of the URL and scope string) as well as a flag indicating whether it should be used in the Datacite test service or the production service.

Listing 16: Example Input

```
{
    "as_test": true,
    "username": "DATACITE_USERNAME",
    "password": "DATACITE_PASSWORD",
    "Doi": {
        "id": "10.80206/ap_test",
```

(continues on next page)

(continued from previous page)

```
        "type": "dois",
        "attributes": {
            "doi": "10.80206/ap_test",
            "creators": [
                {
                    "name": "Globus Dev Team"
                }
            ],
            "titles": [
                {
                    "title": "Test Title"
                }
            ],
            "publisher": "Globus",
            "publicationYear": "2020"
        }
    }
}
```

CHAPTER

EIGHT

CLI REFERENCE

CHAPTER

NINE

GLOBUS-AUTOMATE

CLI for Globus Automate

By default, this CLI keeps all its config and cached tokens in .globus_automate_tokens.json in the user's home directory.

Usage:

```
$ globus-automate [OPTIONS] COMMAND [ARGS]...
```

Options:

- `-V, --version`: Print CLI version number and exit
- `--install-completion`: Install completion for the current shell.
- `--show-completion`: Show completion for the current shell, to copy it or customize the installation.
- `--help`: Show this message and exit.

Commands:

- `action`: Manage Globus Automate Actions
- `flow`: Manage Globus Automate Flows
- `queue`: Manage Globus Automate Queues
- `session`: Manage your session with the Automate Command Line

9.1 globus-automate action

Usage:

```
$ globus-automate action [OPTIONS] COMMAND [ARGS]...
```

Options:

- `--help`: Show this message and exit.

Commands:

- `cancel`: Terminate a running Action by its ACTION_ID.
- `introspect`: Introspect an Action Provider's schema.
- `release`: Remove an Action's execution history by its…
- `resume`: Resume an inactive Action by its ACTION_ID.
- `run`: Launch an Action.

- `status`: Query an Action's status by its ACTION_ID.

9.1.1 globus-automate action cancel

Terminate a running Action by its ACTION_ID.

Usage:

```
$ globus-automate action cancel [OPTIONS] ACTION_ID
```

Arguments:

- `ACTION_ID`: [required]

Options:

- `--action-url TEXT`: The url at which the target Action Provider is located. [required]
- `--action-scope TEXT`: The scope this Action Provider uses to authenticate requests.
- `-v, --verbose`: Run with increased verbosity
- `-f, --format [json|yaml]`: Output display format. [default: json]
- `--help`: Show this message and exit.

9.1.2 globus-automate action introspect

Introspect an Action Provider's schema.

Usage:

```
$ globus-automate action introspect [OPTIONS]
```

Options:

- `--action-url TEXT`: The url at which the target Action Provider is located. [required]
- `--action-scope TEXT`: The scope this Action Provider uses to authenticate requests.
- `-v, --verbose`: Run with increased verbosity
- `-f, --format [json|yaml]`: Output display format. [default: json]
- `--help`: Show this message and exit.

9.1.3 globus-automate action release

Remove an Action's execution history by its ACTION_ID.

Usage:

```
$ globus-automate action release [OPTIONS] ACTION_ID
```

Arguments:

- `ACTION_ID`: [required]

Options:

- `--action-url TEXT`: The url at which the target Action Provider is located. [required]

- `--action-scope TEXT`: The scope this Action Provider uses to authenticate requests.
- `-v, --verbose`: Run with increased verbosity
- `-f, --format [json|yaml]`: Output display format. [default: json]
- `--help`: Show this message and exit.

9.1.4 globus-automate action resume

Resume an inactive Action by its ACTION_ID.

Usage:

```
$ globus-automate action resume [OPTIONS] ACTION_ID
```

Arguments:

- `ACTION_ID`: [required]

Options:

- `--action-url TEXT`: The url at which the target Action Provider is located. [required]
- `--action-scope TEXT`: The scope this Action Provider uses to authenticate requests.
- `--query-for-inactive-reason / --no-query-for-inactive-reason`: Should the Action first be queried to determine the reason for the resume, and prompt for additional consent if needed. [default: True]
- `-v, --verbose`: Run with increased verbosity
- `-f, --format [json|yaml]`: Output display format. [default: json]
- `-w, --watch`: Continuously poll this Action until it reaches a completed state. [default: False]
- `--help`: Show this message and exit.

9.1.5 globus-automate action run

Launch an Action.

Usage:

```
$ globus-automate action run [OPTIONS]
```

Options:

- `--action-url TEXT`: The url at which the target Action Provider is located. [required]
- `--action-scope TEXT`: The scope this Action Provider uses to authenticate requests.
- `-b, --body TEXT`: The body to supply to the Action Provider. Can be a filename or raw JSON string. [required]
- `--request-id TEXT`: An identifier to associate with this Action invocation request
- `--manage-by TEXT`: A principal which may change the execution of the Action. The principal is the user's or group's UUID prefixed with either 'urn:globus:groups:id:' or 'urn:globus:auth:identity:' [repeatable]
- `--monitor-by TEXT`: A principal which may view the state of the Action. The principal is the user's or group's UUID prefixed with either 'urn:globus:groups:id:' or 'urn:globus:auth:identity:' [repeatable]
- `-v, --verbose`: Run with increased verbosity
- `-f, --format [json|yaml]`: Output display format. [default: json]

- `-l, --label TEXT`: Optional label to mark this execution of the action.
- `-w, --watch`: Continuously poll this Action until it reaches a completed state. [default: False]
- `--help`: Show this message and exit.

9.1.6 globus-automate action status

Query an Action's status by its ACTION_ID.

Usage:

```
$ globus-automate action status [OPTIONS] ACTION_ID
```

Arguments:

- `ACTION_ID`: [required]

Options:

- `--action-url TEXT`: The url at which the target Action Provider is located. [required]
- `--action-scope TEXT`: The scope this Action Provider uses to authenticate requests.
- `-v, --verbose`: Run with increased verbosity
- `-f, --format [json|yaml]`: Output display format. [default: json]
- `-w, --watch`: Continuously poll this Action until it reaches a completed state. [default: False]
- `--help`: Show this message and exit.

9.2 globus-automate flow

Manage Globus Automate Flows

To target a different Flows service endpoint, export the GLOBUS_AUTOMATE_FLOWS_ENDPOINT environment variable.

Usage:

```
$ globus-automate flow [OPTIONS] COMMAND [ARGS]...
```

Options:

- `--help`: Show this message and exit.

Commands:

- `action-cancel`: Cancel an active execution for a particular…
- `action-enumerate`: Retrieve all Flow Runs you have access to…
- `action-list`: List a Flow definition's discrete…
- `action-log`: Get a log of the steps executed by a Flow…
- `action-release`: Remove execution history for a particular…
- `action-resume`: Resume a Flow in the INACTIVE state.
- `action-status`: Display the status for a Flow definition's…

- `action-update`: Update a Run on the Flows service.
- `batch-run-update`: Update metadata and permissions on one or…
- `delete`: Delete a Flow.
- `deploy`: Deploy a new Flow.
- `display`: Visualize a local or deployed Flow…
- `get`: Get a Flow's definition as it exists on the…
- `lint`: Parse and validate a Flow definition by…
- `list`: List Flows for which you have access.
- `run`: Run an instance of a Flow.
- `run-cancel`: Cancel an active execution for a particular…
- `run-enumerate`: Retrieve all Flow Runs you have access to…
- `run-list`: List a Flow definition's discrete…
- `run-log`: Get a log of the steps executed by a Flow…
- `run-release`: Remove execution history for a particular…
- `run-resume`: Resume a Flow in the INACTIVE state.
- `run-status`: Display the status for a Flow definition's…
- `run-update`: Update a Run on the Flows service.
- `update`: Update a Flow.

9.2.1 globus-automate flow action-cancel

Cancel an active execution for a particular Flow definition's invocation.

Usage:

```
$ globus-automate flow action-cancel [OPTIONS] ACTION_ID
```

Arguments:

- `ACTION_ID`: [required]

Options:

- `--flow-id TEXT`: The ID for the Flow which triggered the Action. [required]
- `--flow-scope TEXT`: The scope this Flow uses to authenticate requests.
- `-f, --format [json|yaml]`: Output display format. [default: json]
- `-v, --verbose`: Run with increased verbosity
- `--help`: Show this message and exit.

9.2.2 globus-automate flow action-enumerate

Retrieve all Flow Runs you have access to view.

Usage:

```
$ globus-automate flow action-enumerate [OPTIONS]
```

Options:

- `--role [run_monitor|run_manager|run_owner|created_by|monitor_by|manage_by]`: Display Actions/Runs where you have at least the selected role. Precedence of roles is: run_monitor, run_manager, run_owner. Thus, by specifying, for example, run_manager, all flows for which you have run_manager or run_owner roles will be displayed. Values monitored_by, managed_by and created_by are deprecated. [repeatable use deprecated as the lowest precedence value provided will determine the Actions/Runs displayed.] [default: ActionRole.run_owner]
- `--status [SUCCEEDED|FAILED|ACTIVE|INACTIVE]`: Display Actions with the selected status. [repeatable] [default:]
- `-m, --marker TEXT`: A pagination token for iterating through returned data.
- `-p, --per-page INTEGER RANGE`: The page size to return. Only valid when used without providing a marker.
- `--filter TEXT`: A filtering criteria in the form 'key=value' to apply to the resulting Action listing. The key indicates the filter, the value indicates the pattern to match. Multiple patterns for a single key may be specified as a comma separated string, the results for which will represent a logical OR. If multiple filters are applied, the returned data will be the result of a logical AND between them. [repeatable]
- `--orderby TEXT`: An ordering criteria in the form 'key=value' to apply to the resulting Flow listing. The key indicates the field to order on, and the value is either ASC, for ascending order, or DESC, for descending order. The first ordering criteria will be used to sort the data, subsequent ordering criteria will further sort ties. [repeatable]
- `-w, --watch`: Continuously poll for new Actions. [default: False]
- `-f, --format [json|yaml|table]`: Output display format. [default: table]
- `-v, --verbose`: Run with increased verbosity
- `--help`: Show this message and exit.

9.2.3 globus-automate flow action-list

List a Flow definition's discrete invocations.

Usage:

```
$ globus-automate flow action-list [OPTIONS]
```

Options:

- `--flow-id TEXT`: The ID for the Flow which triggered the Action. If not present runs from all Flows will be displayed.
- `--flow-scope TEXT`: The scope this Flow uses to authenticate requests.
- `--role [run_monitor|run_manager|run_owner|created_by|monitor_by|manage_by]`: Display Actions/Runs where you have at least the selected role. Precedence of roles is: run_monitor, run_manager, run_owner. Thus, by specifying, for example, run_manager, all runs for which you have run_manager or

run_owner roles will be displayed. [repeatable use deprecated as the lowest precedence value provided will determine the flows displayed.]

- `--status [SUCCEEDED|FAILED|ACTIVE|INACTIVE]`: Display Actions with the selected status. [repeatable] [default:]
- `-m, --marker TEXT`: A pagination token for iterating through returned data.
- `-p, --per-page INTEGER RANGE`: The page size to return. Only valid when used without providing a marker.
- `--filter TEXT`: A filtering criteria in the form 'key=value' to apply to the resulting Action listing. The key indicates the filter, the value indicates the pattern to match. Multiple patterns for a single key may be specified as a comma separated string, the results for which will represent a logical OR. If multiple filters are applied, the returned data will be the result of a logical AND between them. [repeatable]
- `--orderby TEXT`: An ordering criteria in the form 'key=value' to apply to the resulting Flow listing. The key indicates the field to order on, and the value is either ASC, for ascending order, or DESC, for descending order. The first ordering criteria will be used to sort the data, subsequent ordering criteria will further sort ties. [repeatable]
- `-v, --verbose`: Run with increased verbosity
- `-w, --watch`: Continuously poll for new Actions. [default: False]
- `-f, --format [json|yaml|table]`: Output display format. [default: table]
- `--help`: Show this message and exit.

9.2.4 globus-automate flow action-log

Get a log of the steps executed by a Flow definition's invocation.

Usage:

```
$ globus-automate flow action-log [OPTIONS] ACTION_ID
```

Arguments:

- `ACTION_ID`: [required]

Options:

- `--flow-id TEXT`: The ID for the Flow which triggered the Action. [required]
- `--flow-scope TEXT`: The scope this Flow uses to authenticate requests.
- `--reverse`: Display logs starting from most recent and proceeding in reverse chronological order [default: False]
- `--limit INTEGER RANGE`: Set a maximum number of events from the log to return
- `-m, --marker TEXT`: A pagination token for iterating through returned data.
- `-p, --per-page INTEGER RANGE`: The page size to return. Only valid when used without providing a marker.
- `-f, --format [json|yaml|table|image|graphiz]`: Output display format. [default: table]
- `-w, --watch`: Continuously poll this Action until it reaches a completed state. Using this option will report only the latest state available. [default: False]
- `-v, --verbose`: Run with increased verbosity
- `--help`: Show this message and exit.

9.2.5 globus-automate flow action-release

Remove execution history for a particular Flow definition's invocation. After this, no further information about the run can be accessed.

Usage:

```
$ globus-automate flow action-release [OPTIONS] ACTION_ID
```

Arguments:

- `ACTION_ID`: [required]

Options:

- `--flow-id TEXT`: The ID for the Flow which triggered the Action. [required]
- `--flow-scope TEXT`: The scope this Flow uses to authenticate requests.
- `-f, --format [json|yaml]`: Output display format. [default: json]
- `-v, --verbose`: Run with increased verbosity
- `--help`: Show this message and exit.

9.2.6 globus-automate flow action-resume

Resume a Flow in the INACTIVE state. If query-for-inactive-reason is set, and the Flow Action is in an INACTIVE state due to requiring additional Consent, the required Consent will be determined and you may be prompted to allow Consent using the Globus Auth web interface.

Usage:

```
$ globus-automate flow action-resume [OPTIONS] ACTION_ID
```

Arguments:

- `ACTION_ID`: [required]

Options:

- `--flow-id TEXT`: The ID for the Flow which triggered the Action. [required]
- `--flow-scope TEXT`: The scope this Flow uses to authenticate requests.
- `--query-for-inactive-reason / --no-query-for-inactive-reason`: Should the Action first be queried to determine the reason for the resume, and prompt for additional consent if needed. [default: True]
- `-f, --format [json|yaml]`: Output display format. [default: json]
- `-w, --watch`: Continuously poll this Action until it reaches a completed state. [default: False]
- `-v, --verbose`: Run with increased verbosity
- `--help`: Show this message and exit.

9.2.7 globus-automate flow action-status

Display the status for a Flow definition's particular invocation.

Usage:

```
$ globus-automate flow action-status [OPTIONS] ACTION_ID
```

Arguments:

- `ACTION_ID`: [required]

Options:

- `--flow-id UUID`: The ID for the Flow which triggered the Action. [required]
- `--flow-scope TEXT`: The scope this Flow uses to authenticate requests.
- `-w, --watch`: Continuously poll this Action until it reaches a completed state. [default: False]
- `-f, --format [json|yaml]`: Output display format. [default: json]
- `-v, --verbose`: Run with increased verbosity
- `--help`: Show this message and exit.

9.2.8 globus-automate flow action-update

Update a Run on the Flows service.

Usage:

```
$ globus-automate flow action-update [OPTIONS] RUN_ID
```

Arguments:

- `RUN_ID`: [required]

Options:

- `--run-manager TEXT`: A principal which may change the execution of the Run.The principal value is the user's Globus Auth username or their identity UUID in the form urn:globus:auth:identity:. A Globus Group may also be used using the form urn:globus:groups:id:. Specify an empty string once to erase all Run managers. [repeatable]
- `--run-monitor TEXT`: A principal which may monitor the execution of the Run.The principal value is the user's Globus Auth username or their identity UUID in the form urn:globus:auth:identity:. A Globus Group may also be used using the form urn:globus:groups:id:. [repeatable]
- `--tag TEXT`: A tag to associate with the Run. If specified, the existing tags on the Run will be replaced with the list of tags specified here. Specify an empty string once to erase all tags. [repeatable]
- `--label TEXT`: A label to associate with the Run.
- `-v, --verbose`: Run with increased verbosity
- `-f, --format [json|yaml]`: Output display format. [default: json]
- `--help`: Show this message and exit.

9.2.9 globus-automate flow batch-run-update

Update metadata and permissions on one or more Runs.

Modifying lists of values ========================

Most options support set, add, and remove operations.

The "add" option variants will add the specified value to whatever is set on each affected Run. For example, if one Run has a "star" tag and another has a "circle" tag, `--add-tag square` will result in a Run with "star" and "square" tags, and the other Run will have "circle" and "square" tags.

The "remove" option variants will remove the specified value from whatever is set on each affected Run. There will not be an error if the value is not set on a Run. For example, if one Run has a "star" tag and another has a "circle" tag, `--remove-tag star` will result in a Run with no tags while the other still has a "circle" tag.

The "set" option variants will overwrite the metadata and permissions currently set on all affected Runs. For example, `--set-tag example` will standardize all affected Runs so that they have just one tag: "example".

To remove all values on all affected Runs, use the "set" variant of an option with an empty string. For example, to erase all Run monitors, use `--set-run-monitors ""`.

All options with "set", "add", and "remove" variants can be used multiple times. However, only one variation of an option can be specified at a time. For example, `--set-tag` and `--add-tag` cannot be combined in the same command, and `--set-run-manager` and `--add-run-manager` cannot be combined. It is fine to combine `--add-tag` and `--remove-run-manager`.

Modifying roles ===============

Run managers and monitors must be specified in one of these forms:

* A user's Globus Auth username * A user's identity UUID in the form urn:globus:auth:identity: * A group's identity UUID in the form urn:globus:groups:id:

Usage:

```
$ globus-automate flow batch-run-update [OPTIONS] RUN_IDS...
```

Arguments:

- `RUN_IDS...`: [required]

Options:

- `--set-run-manager TEXT`: Set a principal on affected Runs that can change the Run execution.
- `--add-run-manager TEXT`: Add a principal to affected Runs that can change the Run execution.
- `--remove-run-manager TEXT`: Remove a principal from affected Runs that can change the Run execution.
- `--set-run-monitor TEXT`: Set a principal on affected Runs that can monitor Run execution.
- `--add-run-monitor TEXT`: Add a principal to affected Runs that can monitor Run execution.
- `--remove-run-monitor TEXT`: Remove a principal from affected Runs that can monitor Run execution.
- `--set-tag TEXT`: A tag to set on the specified Runs.
- `--add-tag TEXT`: A tag to add to the affected Runs.
- `--remove-tag TEXT`: A tag to remove from the affected Runs.
- `--status TEXT`: Set the status of the affected Runs.

Currently, "cancel" is the only valid value. * `-v, --verbose`: Run with increased verbosity * `-f, --format [json|yaml]`: Output display format. [default: json] * `--help`: Show this message and exit.

9.2.10 globus-automate flow delete

Delete a Flow. You must be in the Flow's "flow_administrators" list.

Usage:

```
$ globus-automate flow delete [OPTIONS] FLOW_ID
```

Arguments:

- `FLOW_ID`: [required]

Options:

- `-f, --format [json|yaml]`: Output display format. [default: json]
- `-v, --verbose`: Run with increased verbosity
- `--help`: Show this message and exit.

9.2.11 globus-automate flow deploy

Deploy a new Flow.

Usage:

```
$ globus-automate flow deploy [OPTIONS]
```

Options:

- `--title TEXT`: The Flow's title. [required]
- `--definition TEXT`: JSON or YAML representation of the Flow to deploy. May be provided as a filename or a raw string representing a JSON object or YAML definition. [required]
- `--subtitle TEXT`: A subtitle for the Flow providing additional, brief description.
- `--description TEXT`: A long form description of the Flow's purpose or usage.
- `--input-schema TEXT`: A JSON or YAML representation of a JSON Schema which will be used to validate the input to the deployed Flow when it is run. If not provided, no validation will be performed on Flow input. May be provided as a filename or a raw string.
- `--keyword TEXT`: A keyword which may categorize or help discover the Flow. [repeatable]
- `--flow-viewer TEXT`: A principal which may view this Flow. The principal value is the user's Globus Auth username or their identity UUID in the form urn:globus:auth:identity:. A Globus Group may also be used using the form urn:globus:groups:id:. The special value of 'public' may be used to indicate that any user can view this Flow. [repeatable]
- `--flow-starter TEXT`: A principal which may run an instance of the deployed Flow. The principal value is the user's Globus Auth username or their identity UUID in the form urn:globus:auth:identity:. A Globus Group may also be used using the form urn:globus:groups:id:.The special value of 'all_authenticated_users' may be used to indicate that any authenticated user can invoke this flow. [repeatable]
- `--flow-administrator TEXT`: A principal which may update the deployed Flow. The principal value is the user's Globus Auth username or their identity UUID in the form urn:globus:auth:identity:. A Globus Group may also be used using the form urn:globus:groups:id:.{[}repeatable]
- `--subscription-id TEXT`: The Id of the Globus Subscription which will be used to make this flow managed.

- `--validate / --no-validate`: (EXPERIMENTAL) Perform rudimentary validation of the flow definition. [default: True]
- `-v, --verbose`: Run with increased verbosity
- `-f, --format [json|yaml]`: Output display format. [default: json]
- `--dry-run`: Do a dry run of deploying the flow to test your definition without actually making changes. [default: False]
- `--help`: Show this message and exit.

9.2.12 globus-automate flow display

Visualize a local or deployed Flow definition. If providing a Flow's ID, You must have either created the Flow or be present in the Flow's "flow_viewers" list to view it.

Usage:

```
$ globus-automate flow display [OPTIONS] [FLOW_ID]
```

Arguments:

- `[FLOW_ID]`

Options:

- `--flow-definition TEXT`: JSON or YAML representation of the Flow to display. May be provided as a filename or a raw string representing a JSON object or YAML definition.
- `-f, --format [json|yaml|image|graphviz]`: Output display format. [default: json]
- `--help`: Show this message and exit.

9.2.13 globus-automate flow get

Get a Flow's definition as it exists on the Flows service.

Usage:

```
$ globus-automate flow get [OPTIONS] FLOW_ID
```

Arguments:

- `FLOW_ID`: A deployed Flow's ID [required]

Options:

- `-f, --format [json|yaml]`: Output display format. [default: json]
- `-v, --verbose`: Run with increased verbosity
- `--help`: Show this message and exit.

9.2.14 globus-automate flow lint

Parse and validate a Flow definition by providing visual output.

Usage:

```
$ globus-automate flow lint [OPTIONS]
```

Options:

- `--definition TEXT`: JSON or YAML representation of the Flow to deploy. May be provided as a filename or a raw string. [required]
- `--help`: Show this message and exit.

9.2.15 globus-automate flow list

List Flows for which you have access.

Usage:

```
$ globus-automate flow list [OPTIONS]
```

Options:

- `-r, --role [flow_viewer|flow_starter|flow_administrator|flow_owner|created_by|visible_to|runnable_b`
  Display Flows where you have at least the selected role. Precedence of roles is: flow_viewer, flow_starter, flow_administrator, flow_owner. Thus, by specifying, for example, flow_starter, all flows for which you have flow_starter, flow_administrator, or flow_owner roles will be displayed. Values visible_to, runnable_by, administered_by and created_by are deprecated. [repeatable use deprecated as the lowest precedence value provided will determine the flows displayed.] [default: FlowRole.flow_owner]
- `-m, --marker TEXT`: A pagination token for iterating through returned data.
- `-p, --per-page INTEGER RANGE`: The page size to return. Only valid when used without providing a marker.
- `--filter TEXT`: A filtering criteria in the form 'key=value' to apply to the resulting Flow listing. The key indicates the filter, the value indicates the pattern to match. Multiple patterns for a single key may be specified as a comma separated string, the results for which will represent a logical OR. If multiple filters are applied, the returned data will be the result of a logical AND between them. [repeatable]
- `--orderby TEXT`: An ordering criteria in the form 'key=value' to apply to the resulting Flow listing. The key indicates the field to order on, and the value is either ASC, for ascending order, or DESC, for descending order. The first ordering criteria will be used to sort the data, subsequent ordering criteria will further sort ties. [repeatable]
- `-v, --verbose`: Run with increased verbosity
- `-f, --format [json|yaml|table]`: Output display format. [default: table]
- `-w, --watch`: Continuously poll for new Flows. [default: False]
- `--help`: Show this message and exit.

9.2.16 globus-automate flow run

Run an instance of a Flow. The argument provides the initial state of the Flow. You must be in the Flow's "flow_starters" list.

Usage:

```
$ globus-automate flow run [OPTIONS] FLOW_ID
```

Arguments:

- `FLOW_ID`: [required]

Options:

- `--flow-input TEXT`: JSON or YAML formatted input to the Flow. May be provided as a filename or a raw string. [required]
- `--flow-scope TEXT`: The scope this Flow uses to authenticate requests.
- `--run-manager TEXT`: A principal which may change the execution of the Flow instance. The principal value is the user's Globus Auth username or their identity UUID in the form urn:globus:auth:identity:. A Globus Group may also be used using the form urn:globus:groups:id:. [repeatable]
- `--run-monitor TEXT`: A principal which may monitor the execution of the Flow instance. The principal value is the user's Globus Auth username or their identity UUID in the form urn:globus:auth:identity:. A Globus Group may also be used using the form urn:globus:groups:id:. [repeatable]
- `-v, --verbose`: Run with increased verbosity
- `-f, --format [json|yaml|table]`: Output display format. If –watch is enabled then the default is 'table', otherwise 'json' is the default.
- `-l, --label TEXT`: Label to mark this run. [required]
- `--tag TEXT`: A tag to associate with this Run.

This option can be used multiple times. The full collection of tags will associated with the Run. * `-w, --watch`: Continuously poll this Action until it reaches a completed state. If enabled the default output format is 'table'. [default: False] * `--dry-run`: Do a dry run with your input to this flow to test the input without actually running anything. [default: False] * `--help`: Show this message and exit.

9.2.17 globus-automate flow run-cancel

Cancel an active execution for a particular Flow definition's invocation.

Usage:

```
$ globus-automate flow run-cancel [OPTIONS] ACTION_ID
```

Arguments:

- `ACTION_ID`: [required]

Options:

- `--flow-id TEXT`: The ID for the Flow which triggered the Action. [required]
- `--flow-scope TEXT`: The scope this Flow uses to authenticate requests.
- `-f, --format [json|yaml]`: Output display format. [default: json]
- `-v, --verbose`: Run with increased verbosity

- `--help`: Show this message and exit.

9.2.18 globus-automate flow run-enumerate

Retrieve all Flow Runs you have access to view.

Usage:

```
$ globus-automate flow run-enumerate [OPTIONS]
```

Options:

- `--role [run_monitor|run_manager|run_owner|created_by|monitor_by|manage_by]`: Display Actions/Runs where you have at least the selected role. Precedence of roles is: run_monitor, run_manager, run_owner. Thus, by specifying, for example, run_manager, all flows for which you have run_manager or run_owner roles will be displayed. Values monitored_by, managed_by and created_by are deprecated. [repeatable use deprecated as the lowest precedence value provided will determine the Actions/Runs displayed.] [default: ActionRole.run_owner]
- `--status [SUCCEEDED|FAILED|ACTIVE|INACTIVE]`: Display Actions with the selected status. [repeatable] [default:]
- `-m, --marker TEXT`: A pagination token for iterating through returned data.
- `-p, --per-page INTEGER RANGE`: The page size to return. Only valid when used without providing a marker.
- `--filter TEXT`: A filtering criteria in the form 'key=value' to apply to the resulting Action listing. The key indicates the filter, the value indicates the pattern to match. Multiple patterns for a single key may be specified as a comma separated string, the results for which will represent a logical OR. If multiple filters are applied, the returned data will be the result of a logical AND between them. [repeatable]
- `--orderby TEXT`: An ordering criteria in the form 'key=value' to apply to the resulting Flow listing. The key indicates the field to order on, and the value is either ASC, for ascending order, or DESC, for descending order. The first ordering criteria will be used to sort the data, subsequent ordering criteria will further sort ties. [repeatable]
- `-w, --watch`: Continuously poll for new Actions. [default: False]
- `-f, --format [json|yaml|table]`: Output display format. [default: table]
- `-v, --verbose`: Run with increased verbosity
- `--help`: Show this message and exit.

9.2.19 globus-automate flow run-list

List a Flow definition's discrete invocations.

Usage:

```
$ globus-automate flow run-list [OPTIONS]
```

Options:

- `--flow-id TEXT`: The ID for the Flow which triggered the Action. If not present runs from all Flows will be displayed.
- `--flow-scope TEXT`: The scope this Flow uses to authenticate requests.

- `--role [run_monitor|run_manager|run_owner|created_by|monitor_by|manage_by]`: Display Actions/Runs where you have at least the selected role. Precedence of roles is: run_monitor, run_manager, run_owner. Thus, by specifying, for example, run_manager, all runs for which you have run_manager or run_owner roles will be displayed. [repeatable use deprecated as the lowest precedence value provided will determine the flows displayed.]
- `--status [SUCCEEDED|FAILED|ACTIVE|INACTIVE]`: Display Actions with the selected status. [repeatable] [default:]
- `-m, --marker TEXT`: A pagination token for iterating through returned data.
- `-p, --per-page INTEGER RANGE`: The page size to return. Only valid when used without providing a marker.
- `--filter TEXT`: A filtering criteria in the form 'key=value' to apply to the resulting Action listing. The key indicates the filter, the value indicates the pattern to match. Multiple patterns for a single key may be specified as a comma separated string, the results for which will represent a logical OR. If multiple filters are applied, the returned data will be the result of a logical AND between them. [repeatable]
- `--orderby TEXT`: An ordering criteria in the form 'key=value' to apply to the resulting Flow listing. The key indicates the field to order on, and the value is either ASC, for ascending order, or DESC, for descending order. The first ordering criteria will be used to sort the data, subsequent ordering criteria will further sort ties. [repeatable]
- `-v, --verbose`: Run with increased verbosity
- `-w, --watch`: Continuously poll for new Actions. [default: False]
- `-f, --format [json|yaml|table]`: Output display format. [default: table]
- `--help`: Show this message and exit.

9.2.20 globus-automate flow run-log

Get a log of the steps executed by a Flow definition's invocation.

Usage:

```
$ globus-automate flow run-log [OPTIONS] ACTION_ID
```

Arguments:

- `ACTION_ID`: [required]

Options:

- `--flow-id TEXT`: The ID for the Flow which triggered the Action. [required]
- `--flow-scope TEXT`: The scope this Flow uses to authenticate requests.
- `--reverse`: Display logs starting from most recent and proceeding in reverse chronological order [default: False]
- `--limit INTEGER RANGE`: Set a maximum number of events from the log to return
- `-m, --marker TEXT`: A pagination token for iterating through returned data.
- `-p, --per-page INTEGER RANGE`: The page size to return. Only valid when used without providing a marker.
- `-f, --format [json|yaml|table|image|graphiz]`: Output display format. [default: table]
- `-w, --watch`: Continuously poll this Action until it reaches a completed state. Using this option will report only the latest state available. [default: False]
- `-v, --verbose`: Run with increased verbosity

- `--help`: Show this message and exit.

9.2.21 globus-automate flow run-release

Remove execution history for a particular Flow definition's invocation. After this, no further information about the run can be accessed.

Usage:

```
$ globus-automate flow run-release [OPTIONS] ACTION_ID
```

Arguments:

- `ACTION_ID`: [required]

Options:

- `--flow-id TEXT`: The ID for the Flow which triggered the Action. [required]
- `--flow-scope TEXT`: The scope this Flow uses to authenticate requests.
- `-f, --format [json|yaml]`: Output display format. [default: json]
- `-v, --verbose`: Run with increased verbosity
- `--help`: Show this message and exit.

9.2.22 globus-automate flow run-resume

Resume a Flow in the INACTIVE state. If query-for-inactive-reason is set, and the Flow Action is in an INACTIVE state due to requiring additional Consent, the required Consent will be determined and you may be prompted to allow Consent using the Globus Auth web interface.

Usage:

```
$ globus-automate flow run-resume [OPTIONS] ACTION_ID
```

Arguments:

- `ACTION_ID`: [required]

Options:

- `--flow-id TEXT`: The ID for the Flow which triggered the Action. [required]
- `--flow-scope TEXT`: The scope this Flow uses to authenticate requests.
- `--query-for-inactive-reason / --no-query-for-inactive-reason`: Should the Action first be queried to determine the reason for the resume, and prompt for additional consent if needed. [default: True]
- `-f, --format [json|yaml]`: Output display format. [default: json]
- `-w, --watch`: Continuously poll this Action until it reaches a completed state. [default: False]
- `-v, --verbose`: Run with increased verbosity
- `--help`: Show this message and exit.

9.2.23 globus-automate flow run-status

Display the status for a Flow definition's particular invocation.

Usage:

```
$ globus-automate flow run-status [OPTIONS] ACTION_ID
```

Arguments:

- `ACTION_ID`: [required]

Options:

- `--flow-id UUID`: The ID for the Flow which triggered the Action. [required]
- `--flow-scope TEXT`: The scope this Flow uses to authenticate requests.
- `-w, --watch`: Continuously poll this Action until it reaches a completed state. [default: False]
- `-f, --format [json|yaml]`: Output display format. [default: json]
- `-v, --verbose`: Run with increased verbosity
- `--help`: Show this message and exit.

9.2.24 globus-automate flow run-update

Update a Run on the Flows service.

Usage:

```
$ globus-automate flow run-update [OPTIONS] RUN_ID
```

Arguments:

- `RUN_ID`: [required]

Options:

- `--run-manager TEXT`: A principal which may change the execution of the Run.The principal value is the user's Globus Auth username or their identity UUID in the form urn:globus:auth:identity:. A Globus Group may also be used using the form urn:globus:groups:id:. Specify an empty string once to erase all Run managers. [repeatable]
- `--run-monitor TEXT`: A principal which may monitor the execution of the Run.The principal value is the user's Globus Auth username or their identity UUID in the form urn:globus:auth:identity:. A Globus Group may also be used using the form urn:globus:groups:id:. [repeatable]
- `--tag TEXT`: A tag to associate with the Run. If specified, the existing tags on the Run will be replaced with the list of tags specified here. Specify an empty string once to erase all tags. [repeatable]
- `--label TEXT`: A label to associate with the Run.
- `-v, --verbose`: Run with increased verbosity
- `-f, --format [json|yaml]`: Output display format. [default: json]
- `--help`: Show this message and exit.

9.2.25 globus-automate flow update

Update a Flow.

Usage:

```
$ globus-automate flow update [OPTIONS] FLOW_ID
```

Arguments:

- `FLOW_ID`: [required]

Options:

- `--title TEXT`: The Flow's title.
- `--definition TEXT`: JSON or YAML representation of the Flow to update. May be provided as a filename or a raw string.
- `--subtitle TEXT`: A subtitle for the Flow providing additional, brief description.
- `--description TEXT`: A long form description of the Flow's purpose or usage.
- `--input-schema TEXT`: A JSON or YAML representation of a JSON Schema which will be used to validate the input to the deployed Flow when it is run. If not provided, no validation will be performed on Flow input. May be provided as a filename or a raw string.
- `--keyword TEXT`: A keyword which may categorize or help discover the Flow. [repeatable]
- `--flow-viewer TEXT`: A principal which may view this Flow. The principal value is the user's Globus Auth username or their identity UUID in the form urn:globus:auth:identity:. A Globus Group may also be used using the form urn:globus:groups:id:.The special value of 'public' may be used to indicate that any user can view this Flow. [repeatable]
- `--flow-starter TEXT`: A principal which may run an instance of the deployed Flow. The principal value is the user's Globus Auth username or their identity UUID in the form urn:globus:auth:identity:. A Globus Group may also be used using the form urn:globus:groups:id:. The special value of 'all_authenticated_users' may be used to indicate that any authenticated user can invoke this flow. [repeatable]
- `--flow-administrator TEXT`: A principal which may update the deployed Flow. The principal value is the user's Globus Auth username or their identity UUID in the form urn:globus:auth:identity:. A Globus Group may also be used using the form urn:globus:groups:id:.{[}repeatable]
- `--assume-ownership`: Assume the ownership of the Flow. This can only be performed by user's in the flow_administrators role. [default: False]
- `--subscription-id TEXT`: The Globus Subscription which will be used to make this flow managed.
- `--validate / --no-validate`: (EXPERIMENTAL) Perform rudimentary validation of the flow definition. [default: True]
- `-v, --verbose`: Run with increased verbosity
- `-f, --format [json|yaml]`: Output display format. [default: json]
- `--help`: Show this message and exit.

9.3 globus-automate queue

Usage:

```
$ globus-automate queue [OPTIONS] COMMAND [ARGS]...
```

Options:

- `--help`: Show this message and exit.

Commands:

- `create`: Create a new Queue.
- `delete`: Delete a Queue based on its id.
- `display`: Display the description of a Queue based on…
- `list`: List Queues for which you have access.
- `message-delete`: Notify a Queue that a message has been…
- `message-receive`: Receive a message from a Queue.
- `message-send`: Send a message to a Queue.
- `update`: Update a Queue's properties.

9.3.1 globus-automate queue create

Create a new Queue.

Usage:

```
$ globus-automate queue create [OPTIONS]
```

Options:

- `--label TEXT`: A convenient name to identify the new Queue. [required]
- `--admin TEXT`: The Principal URNs allowed to administer the Queue. [repeatable] [required]
- `--sender TEXT`: The Principal URNs allowed to send to the Queue. [repeatable] [required]
- `--receiver TEXT`: The Principal URNs allowed to receive from the Queue. [repeatable] [required]
- `--delivery-timeout INTEGER RANGE`: The minimum amount of time (in seconds) that the Queue Service should wait for a message-delete request after delivering a message before making the message visible for receiving by other consumers once again. If used in conjunction with 'receiver_url' this value represents the minimum amount of time (in seconds) that the Queue Service should attempt to retry delivery of messages to the 'receiver_url' if delivery is not initially successful [default: 60]
- `-f, --format [json|yaml]`: Output display format. [default: json]
- `-v, --verbose`: Run with increased verbosity
- `--help`: Show this message and exit.

9.3.2 globus-automate queue delete

Delete a Queue based on its id. You must have either created the Queue or have a role defined on the Queue.

Usage:

```
$ globus-automate queue delete [OPTIONS] QUEUE_ID
```

Arguments:

- QUEUE_ID: [required]

Options:

- -f, --format [json|yaml]: Output display format. [default: json]
- -v, --verbose: Run with increased verbosity
- --help: Show this message and exit.

9.3.3 globus-automate queue display

Display the description of a Queue based on its id.

Usage:

```
$ globus-automate queue display [OPTIONS] QUEUE_ID
```

Arguments:

- QUEUE_ID: [required]

Options:

- -f, --format [json|yaml]: Output display format. [default: json]
- -v, --verbose: Run with increased verbosity
- --help: Show this message and exit.

9.3.4 globus-automate queue list

List Queues for which you have access.

Usage:

```
$ globus-automate queue list [OPTIONS]
```

Options:

- -r, --role [admin|sender|receiver]: Display Queues where you have the selected role. [repeatable] [default: QueueRole.admin]
- -f, --format [json|yaml]: Output display format. [default: json]
- -v, --verbose: Run with increased verbosity
- --help: Show this message and exit.

9.3.5 globus-automate queue message-delete

Notify a Queue that a message has been processed.

Usage:

```
$ globus-automate queue message-delete [OPTIONS] QUEUE_ID
```

Arguments:

- QUEUE_ID: [required]

Options:

- --receipt-handle TEXT: A receipt_handle value returned by a previous call to receive message. [repeatable] [required]
- -f, --format [json|yaml]: Output display format. [default: json]
- -v, --verbose: Run with increased verbosity
- --help: Show this message and exit.

9.3.6 globus-automate queue message-receive

Receive a message from a Queue. You must have the "receiver" role on the Queue to perform this action.

Usage:

```
$ globus-automate queue message-receive [OPTIONS] QUEUE_ID
```

Arguments:

- QUEUE_ID: [required]

Options:

- --max-messages INTEGER RANGE: The maximum number of messages to retrieve from the Queue
- -f, --format [json|yaml]: Output display format. [default: json]
- -v, --verbose: Run with increased verbosity
- --help: Show this message and exit.

9.3.7 globus-automate queue message-send

Send a message to a Queue. You must have the "sender" role on the Queue to perform this action.

Usage:

```
$ globus-automate queue message-send [OPTIONS] QUEUE_ID
```

Arguments:

- QUEUE_ID: [required]

Options:

- -m, --message TEXT: Text of the message to send. Files may also be referenced. [required]
- -f, --format [json|yaml]: Output display format. [default: json]

- `-v, --verbose`: Run with increased verbosity
- `--help`: Show this message and exit.

9.3.8 globus-automate queue update

Update a Queue's properties. Requires the admin role on the Queue.

Usage:

```
$ globus-automate queue update [OPTIONS] QUEUE_ID
```

Arguments:

- `QUEUE_ID`: [required]

Options:

- `--label TEXT`: A convenient name to identify the new Queue. [required]
- `--admin TEXT`: The Principal URNs allowed to administer the Queue. [repeatable] [required]
- `--sender TEXT`: The Principal URNs allowed to send to the Queue. [repeatable] [required]
- `--receiver TEXT`: The Principal URNs allowed to receive from the Queue. [repeatable] [required]
- `--delivery-timeout INTEGER RANGE`: The minimum amount of time (in seconds) that the Queue Service should wait for a message-delete request after delivering a message before making the message visible for receiving by other consumers once again. If used in conjunction with 'receiver_url' this value represents the minimum amount of time (in seconds) that the Queue Service should attempt to retry delivery of messages to the 'receiver_url' if delivery is not initially successful [required]
- `--visibility-timeout INTEGER RANGE`: [default: 30]
- `-f, --format [json|yaml]`: Output display format. [default: json]
- `-v, --verbose`: Run with increased verbosity
- `--help`: Show this message and exit.

9.4 globus-automate session

Usage:

```
$ globus-automate session [OPTIONS] COMMAND [ARGS]...
```

Options:

- `--help`: Show this message and exit.

Commands:

- `logout`: Remove all locally cached Globus Automate…
- `revoke`: Remove all locally cached Globus Automate…
- `whoami`: Determine the username for the identity…

9.4.1 globus-automate session logout

Remove all locally cached Globus Automate authentication information.

Usage:

```
$ globus-automate session logout [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

9.4.2 globus-automate session revoke

Remove all locally cached Globus Automate authentication information and invalidate all locally cached access or refresh tokens. These tokens can no longer be used elsewhere.

Usage:

```
$ globus-automate session revoke [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

9.4.3 globus-automate session whoami

Determine the username for the identity logged in to Globus Auth. If run with increased verbosity, the caller's full user information is displayed.

Usage:

```
$ globus-automate session whoami [OPTIONS]
```

Options:

- `-v, --verbose`: Run with increased verbosity
- `--help`: Show this message and exit.

CHAPTER

TEN

PYTHON SDK REFERENCE

The Globus Automate Client package provides a Python SDK for interfacing with and invoking Globus Actions and Flows. Below, we provide information on helper functions available for creating authenticated Action and Flow clients as well as documentation on the methods available to those clients.

10.1 Actions Client

class globus_automate_client.action_client.**ActionClient**(**args*, ***kwargs*)

Parameters

- **args** (*Any*) –
- **kwargs** (*Any*) –

Return type
Any

property action_scope: str

This property can be used to determine an `ActionClient`'s `action_scope`. Internally, this property will introspect the Action Provider at the URL for which the `ActionClient` was created. If the `Action Provider` is not public, a valid `Globus Authorizer` will have to have been provided on initialization to the `ActionClient`. Otherwise, this call will fail.

introspect(**_)

Introspect the details of an Action Provider to discover information such as its expected `action_scope`, its `input_schema`, and who to contact when there's trouble.

Return type
globus_sdk.GlobusHTTPResponse

run(*body*, *request_id=None*, *manage_by=None*, *monitor_by=None*, *label=None*, *tags=None*, *force_path=None*, ***kwargs*)

Invoke the Action Provider to execute an Action with the given parameters.

Parameters

- **body** (*Mapping[str, Any]*) – The Action Provider specific input required to execute an Action payload
- **request_id** (*Optional[str]*) – An optional identifier that serves to de-duplicate requests to the Action Provider

- **manage_by** (*Optional[Iterable[str]]*) – A series of Globus identities which may alter this Action's execution. The principal value is the user's or group's UUID prefixed with either 'urn:globus:groups:id:' or 'urn:globus:auth:identity:'
- **monitor_by** (*Optional[Iterable[str]]*) – A series of Globus identities which may view the state of this Action. The principal value is the user's or group's UUID prefixed with either 'urn:globus:groups:id:' or 'urn:globus:auth:identity:'
- **force_path** (*Optional[str]*) – A URL to use for running this action, ignoring any previous configuration
- **label** (*Optional[str]*) – Set a label for the Action that is run.
- **tags** (*Optional[List[str]]*) – A list of tags to associate with the Run.
- **run_monitors** – May be used as an alias for `monitor_by`
- **run_managers** – May be used as an alias for `manage_by`

Return type
globus_sdk.GlobusHTTPResponse

status(*action_id*)

Query the Action Provider for the status of executed Action

Parameters
action_id (*str*) – An identifier that uniquely identifies an Action executed on this Action Provider.

Return type
globus_sdk.GlobusHTTPResponse

resume(*action_id*)

Resume an INACTIVE action. Corrective action must have been taken prior to invoking this method, including the possibility of consenting to additional permissions and using tokens issued by those consents when creating this client. These consents would commonly be required when an Action is INACTIVE and shows the code ConsentRequired.

Parameters
action_id (*str*) – An identifier that uniquely identifies an Action executed on this Action Provider.

Return type
globus_sdk.GlobusHTTPResponse

cancel(*action_id*)

Cancel a currently executing Action on an Action Provider

Parameters
action_id (*str*) – An identifier that uniquely identifies an Action executed on this Action Provider.

Return type
globus_sdk.GlobusHTTPResponse

release(*action_id*)

Remove the history of an Action's execution from an Action Provider

Parameters
action_id (*str*) – An identifier that uniquely identifies an Action executed on this Action Provider.

Return type
globus_sdk.GlobusHTTPResponse

log(*action_id*, *limit=10*, *reverse_order=False*, *marker=None*, *per_page=None*)

Retrieve an Action's execution log history. Not all `Action Providers` support this operation.

Parameters

- **action_id** (*str*) – An identifier that uniquely identifies an Action executed on this Action Provider.
- **limit** (*int*) – A integer specifying how many log records to return
- **reverse_order** (*bool*) – Display the Action states in reverse- chronological order
- **marker** (*Optional[str]*) – A pagination_token indicating the page of results to return and how many entries to return. Not all ActionProviders will support this parameter.
- **per_page** (*Optional[int]*) – The number of results to return per page. If supplied a pagination_token, this parameter has no effect. Not all ActionProviders will support this parameter.

Return type
globus_sdk.GlobusHTTPResponse

classmethod new_client(*action_url*, *authorizer*, *http_timeout=10*)

Classmethod to simplify creating an ActionClient. Use this method when attemping to create an ActionClient with pre-existing credentials or authorizers.

Parameters

- **action_url** (*str*) – The url at which the target Action Provider is located.
- **authorizer** (*Optional[globus_sdk.authorizers.GlobusAuthorizer]*) – The authorizer to use for validating requests to the Action Provider.
- **http_timeout** (*int*) – The amount of time to wait for connections to the Action Provider to be made.

Return type
_ActionClient

Examples

```
>>> auth = ...
>>> url = "https://actions.globus.org/hello_world"
>>> ac = ActionClient.new_client(url, auth)
>>> print(ac.run({"echo_string": "Hello from SDK"}))
```

10.2 Flows Client

class globus_automate_client.flows_client.**FlowsClient**(**args*, ***kwargs*)

This is a specialized type of the Globus Auth service's `BaseClient` used to interact with the Globus Flows service. Any keyword arguments given are passed through to the `BaseClient` constructor.

Parameters

- **args** (*Any*) –
- **kwargs** (*Any*) –

Return type
Any

`use_temporary_authorizer`(*authorizer*)

Temporarily swap out the authorizer instance variable.

This is a context manager. Use it like this:

```
authorizer = self._get_authorizer_for_flow(...)
with self.alternate_authorizer(authorizer):
    ...
```

`deploy_flow`(*flow_definition*, *title*, *subtitle=None*, *description=None*, *keywords=()*, *flow_viewers=()*, *flow_starters=()*, *flow_administrators=()*, *subscription_id=None*, *input_schema=None*, *validate_definition=True*, *validate_schema=True*, *dry_run=False*, ***kwargs*)

Deploys a Flow definition to the Flows service, making the Flow available for execution on the Globus Automate Flows Service.

Parameters

- **flow_definition** (*Mapping[str, Any]*) – A mapping corresponding to a Globus Flows definition.
- **title** (*str*) – A simple, human-readable title for the deployed Flow
- **subtitle** (*Optional[str]*) – A longer, more verbose title for the deployed Flow
- **description** (*Optional[str]*) – A long form explanation of the Flow's purpose or usage
- **keywords** (*Iterable[str]*) – A series of words which may help categorize or make the Flow discoverable
- **flow_viewers** (*Iterable[str]*) – A series of Globus identities which may discover and view the Flow definition
- **flow_starters** (*Iterable[str]*) – A series of Globus identities which may run an instance of this Flow
- **flow_administrators** (*Iterable[str]*) – A series of Globus identities which may update this Flow's definition
- **subscription_id** (*Optional[str]*) – The Globus Subscription which will be used to make this flow managed.
- **input_schema** (*Optional[Mapping[str, Any]]*) – A mapping representing the JSONSchema used to validate input to this Flow. If not supplied, no validation will be done on input to this Flow.
- **validate_definition** (*bool*) – Set to `True` to validate the provided `flow_definition` before attempting to deploy the Flow.
- **validate_schema** (*bool*) – Set to `True` to validate the provided `input_schema` before attempting to deploy the Flow.
- **dry_run** (*bool*) – Set to `True` to test whether the Flow can be deployed successfully.

Return type
globus_sdk.GlobusHTTPResponse

`update_flow`(*flow_id*, *flow_definition=None*, *title=None*, *subtitle=None*, *description=None*, *keywords=None*, *flow_viewers=None*, *flow_starters=None*, *flow_administrators=None*, *subscription_id=None*, *input_schema=None*, *validate_definition=True*, *validate_schema=True*, ***kwargs*)

Updates a deployed Flow's definition or metadata. This method will preserve the existing Flow's values for fields which are not submitted as part of the update.

Parameters

- **flow_id** (*str*) – The UUID for the Flow that will be updated
- **flow_definition** (*Optional[Mapping[str, Any]]*) – A mapping corresponding to a Globus Flows definition
- **title** (*Optional[str]*) – A simple, human-readable title for the deployed Flow
- **subtitle** (*Optional[str]*) – A longer, more verbose title for the deployed Flow
- **description** (*Optional[str]*) – A long form explanation of the Flow's purpose or usage
- **keywords** (*Optional[Iterable[str]]*) – A series of words which may help categorize or make the Flow discoverable
- **flow_viewers** (*Optional[Iterable[str]]*) – A series of Globus identities which may discover and view the Flow definition
- **flow_starters** (*Optional[Iterable[str]]*) – A series of Globus identities which may run an instance of this Flow
- **flow_administrators** (*Optional[Iterable[str]]*) – A series of Globus identities which may update this Flow's definition
- **subscription_id** (*Optional[str]*) – The Globus Subscription which will be used to make this flow managed.
- **input_schema** (*Optional[Mapping[str, Any]]*) – A mapping representing the JSONSchema used to validate input to this Flow. If not supplied, no validation will be done on input to this Flow.
- **validate_definition** (*bool*) – Set to `True` to validate the provided `flow_definition` before attempting to update the Flow.
- **validate_schema** (*bool*) – Set to `True` to validate the provided `input_schema` before attempting to update the Flow.

Return type
globus_sdk.GlobusHTTPResponse

get_flow(*flow_id*, ***kwargs*)

Retrieve a deployed Flow's definition and metadata

Parameters
flow_id (*str*) – The UUID identifying the Flow for which to retrieve details

Return type
globus_sdk.GlobusHTTPResponse

list_flows(*roles=None*, *marker=None*, *per_page=None*, *filters=None*, *orderings=None*, *role=None*, ***kwargs*)

Display all deployed Flows for which you have the selected role(s)

Parameters

- **roles** (*Optional[Iterable[str]]*) – Deprecated since version 0.12: Use `role` instead

See description for `role` parameter. Providing multiple roles behaves as if only a single `role` value is provided and displays the equivalent of the most permissive role.

- **role** (*Optional[str]*) – A role value specifying the minimum role-level permission which will be displayed based on the follow precedence of role values:
  - flow_viewer
  - flow_starter
  - flow_administrators
  - flow_owner

  Thus, if, for example, `flow_starter` is specified, flows for which the user has the `flow_starter`, `flow_administrator` or `flow_owner` roles will be returned.
- **marker** (*Optional[str]*) – A pagination_token indicating the page of results to return and how many entries to return. This is created by the Flows service and returned by operations that support pagination.
- **per_page** (*Optional[int]*) – The number of results to return per page. If supplied a pagination_token, this parameter has no effect.
- **filters** (*Optional[dict]*) – A filtering criteria to apply to the resulting Flow listing. The keys indicate the filter, the values indicate the pattern to match. The returned data will be the result of a logical AND between the filters. Patterns may be comma separated to produce the result of a logical OR.
- **orderings** (*Optional[dict]*) – An ordering criteria to apply to the resulting Flow listing. The keys indicate the field to order on, and the value can be either ASC, for ascending order, or DESC, for descending order. The first ordering criteria will be used to sort the data, subsequent ordering criteria will be applied for ties. Note: To ensure orderings are applied in the correct order, use an OrderedDict if trying to apply multiple orderings.

Return type
globus_sdk.GlobusHTTPResponse

delete_flow(*flow_id*, ***kwargs*)

Remove a Flow definition and its metadata from the Flows service

Parameters
flow_id (*str*) – The UUID identifying the Flow to delete

Return type
globus_sdk.GlobusHTTPResponse

scope_for_flow(*flow_id*)

Returns the scope associated with a particular Flow

Parameters
flow_id (*str*) – The UUID identifying the Flow's scope to lookup

Return type
str

run_flow(*flow_id*, *flow_scope*, *flow_input*, *run_managers=None*, *run_monitors=None*, *dry_run=False*, *label=None*, *tags=None*, ***kwargs*)

Run an instance of a deployed Flow with the given input.

Parameters

- **flow_id** (*str*) – The UUID identifying the Flow to run

- **flow_scope** (*Optional[str]*) – The scope associated with the Flow `flow_id`. If not provided, the SDK will attempt to perform an introspection on the Flow to determine its scope automatically
- **flow_input** (*Mapping*) – A Flow-specific dictionary specifying the input required for the Flow to run.
- **run_managers** (*Optional[Iterable[str]]*) – A series of Globus identities which may alter this Flow instance's execution. The principal value is the user's or group's UUID prefixed with either 'urn:globus:groups:id:' or 'urn:globus:auth:identity:'
- **run_monitors** (*Optional[Iterable[str]]*) – A series of Globus identities which may view this Flow instance's execution state. The principal value is the user's or group's UUID prefixed with either 'urn:globus:groups:id:' or 'urn:globus:auth:identity:'
- **label** (*Optional[str]*) – An optional label which can be used to identify this run
- **tags** (*Optional[List[str]]*) – Tags that will be associated with this Run.
- **kwargs** – Any additional kwargs passed into this method are passed onto the Globus BaseClient. If there exists an "authorizer" keyword argument, that gets used to run the Flow operation. Otherwise, the authorizer_callback defined for the FlowsClient will be used.
- **dry_run** (*bool*) – Set to `True` to test what will happen if the Flow is run without actually running the Flow.

Return type
globus_sdk.GlobusHTTPResponse

flow_action_status(*flow_id*, *flow_scope*, *flow_action_id*, ***kwargs*)

Determine the status for an Action that was launched by a Flow

Parameters

- **flow_id** (*str*) – The UUID identifying the Flow which triggered the Action
- **flow_scope** (*Optional[str]*) – The scope associated with the Flow `flow_id`. If not provided, the SDK will attempt to perform an introspection on the Flow to determine its scope automatically
- **flow_action_id** (*str*) – The ID specifying which Action's status to query
- **kwargs** – Any additional kwargs passed into this method are passed onto the Globus BaseClient. If there exists an "authorizer" keyword argument, that gets used to run the Flow operation. Otherwise, the authorizer_callback defined for the FlowsClient will be used.

Return type
globus_sdk.GlobusHTTPResponse

flow_action_resume(*flow_id*, *flow_scope*, *flow_action_id*, ***kwargs*)

Resume a Flow Action which is in an INACTIVE state.

Parameters

- **flow_id** (*str*) – The UUID identifying the Flow which triggered the Action
- **flow_scope** (*Optional[str]*) – The scope associated with the Flow `flow_id`. If not provided, the SDK will attempt to perform an introspection on the Flow to determine its scope automatically.
- **flow_action_id** (*str*) – The ID specifying the Action with an INACTIVE status we want to resume.

- **kwargs** – Any additional kwargs passed into this method are passed onto the Globus BaseClient. If there exists an "authorizer" keyword argument, that gets used to run the Flow operation. Otherwise, the authorizer_callback defined for the FlowsClient will be used.

Return type
globus_sdk.GlobusHTTPResponse

flow_action_release(*flow_id*, *flow_scope*, *flow_action_id*, ***kwargs*)

Remove the execution history for an Action that was launched by a Flow

Parameters

- **flow_id** (*str*) – The UUID identifying the Flow which launched the Action
- **flow_scope** (*Optional[str]*) – The scope associated with the Flow `flow_id`. If not provided, the SDK will attempt to perform an introspection on the Flow to determine its scope automatically
- **flow_action_id** (*str*) – The ID specifying the Action to release
- **kwargs** – Any additional kwargs passed into this method are passed onto the Globus BaseClient. If there exists an "authorizer" keyword argument, that gets used to run the Flow operation. Otherwise, the authorizer_callback defined for the FlowsClient will be used.

Return type
globus_sdk.GlobusHTTPResponse

flow_action_cancel(*flow_id*, *flow_scope*, *flow_action_id*, ***kwargs*)

Cancel the execution of an Action that was launched by a Flow

Parameters

- **flow_id** (*str*) – The UUID identifying the Flow which launched the Action
- **flow_scope** (*Optional[str]*) – The scope associated with the Flow `flow_id`. If not provided, the SDK will attempt to perform an introspection on the Flow to determine its scope automatically
- **flow_action_id** (*str*) – The ID specifying the Action we want to cancel
- **kwargs** – Any additional kwargs passed into this method are passed onto the Globus BaseClient. If there exists an "authorizer" keyword argument, that gets used to run the Flow operation. Otherwise, the authorizer_callback defined for the FlowsClient will be used.

Return type
globus_sdk.GlobusHTTPResponse

enumerate_runs(*roles=None*, *statuses=None*, *marker=None*, *per_page=None*, *filters=None*, *orderings=None*, *role=None*, ***kwargs*)

Retrieve a listing of Runs the caller has access to. This operation requires the supplied Authorizer to have the RUN_STATUS_SCOPE.

Parameters

- **statuses** (*Optional[Iterable[str]]*) – A list of statuses used to filter the Actions that are returned by the listing. Returned Actions are guaranteed to have one of the specified `statuses`. Valid values are:
  - SUCCEEDED
  - FAILED
  - ACTIVE

- INACTIVE

- **roles** (*Optional[Iterable[str]]*) – Deprecated since version 0.12: Use `role` instead

  See description for `role` parameter. Providing multiple roles behaves as if only a single `role` value is provided and displays the equivalent of the most permissive role.

- **role** (*Optional[str]*) – A role value specifying the minimum role-level permission on the runs which will be returned based on the follow precedence of role values:

  - run_monitor
  - run_manager
  - run_owner

  Thus, if, for example, `run_manager` is specified, runs for which the user has the `run_manager`, or `run_owner` roles will be returned.

- **marker** (*Optional[str]*) – A pagination_token indicating the page of results to return and how many entries to return. This is created by the Flows service and returned by operations that support pagination.

- **per_page** (*Optional[int]*) – The number of results to return per page. If supplied a pagination_token, this parameter has no effect.

- **filters** (*Optional[dict]*) – A filtering criteria to apply to the resulting Action listing. The keys indicate the filter, the values indicate the pattern to match. The returned data will be the result of a logical AND between the filters. Patterns may be comma separated to produce the result of a logical OR.

- **orderings** (*Optional[dict]*) – An ordering criteria to apply to the resulting Action listing. The keys indicate the field to order on, and the value can be either ASC, for ascending order, or DESC, for descending order. The first ordering criteria will be used to sort the data, subsequent ordering criteria will be applied for ties. Note: To ensure orderings are applied in the correct order, use an OrderedDict if trying to apply multiple orderings.

Return type
globus_sdk.GlobusHTTPResponse

`enumerate_actions`(***kwargs*)

An alias for `enumerate_runs`

Return type
globus_sdk.GlobusHTTPResponse

`list_flow_actions`(***kwargs*)

An alias for `list_flow_runs`

Return type
globus_sdk.GlobusHTTPResponse

`list_flow_runs`(*flow_id=None*, *flow_scope=None*, *statuses=None*, *roles=None*, *marker=None*, *per_page=None*, *filters=None*, *orderings=None*, *role=None*, ***kwargs*)

List all Runs for a particular Flow.

If no flow_id is provided, all runs for all Flows will be returned.

Parameters

- **flow_id** (*Optional[str]*) – The UUID identifying the Flow which launched the Run. If not provided, all runs will be returned regardless of which Flow was used to start the Run (equivalent to `enumerate_runs`).

- **flow_scope** (*Optional[str]*) – The scope associated with the Flow flow_id. If not provided, the SDK will attempt to perform an introspection on the Flow to determine its scope automatically
- **statuses** (*Optional[Iterable[str]]*) – The same as in enumerate_runs.
- **roles** (*Optional[Iterable[str]]*) – Deprecated since version 0.12: Use role instead

  The same as in enumerate_runs.
- **marker** (*Optional[str]*) – The same as in enumerate_runs.
- **per_page** (*Optional[int]*) – The same as in enumerate_runs.
- **filters** (*Optional[dict]*) – The same as in enumerate_runs.
- **orderings** (*Optional[dict]*) – The same as in enumerate_runs.
- **role** (*Optional[str]*) – The same as in enumerate_runs.
- **kwargs** – Any additional kwargs passed into this method are passed onto the Globus BaseClient. If there exists an "authorizer" keyword argument, that gets used to run the Flow operation. Otherwise, the authorizer_callback defined for the FlowsClient will be used.

Return type
globus_sdk.GlobusHTTPResponse

flow_action_update(*action_id*, *run_managers=None*, *run_monitors=None*, *tags=None*, *label=None*, ***kwargs*)

Update a Flow Action.

Parameters

- **action_id** (*str*) – The UUID identifying the Action to update
- **run_managers** (*Optional[Sequence[str]]*) – A list of Globus Auth URNs which will have the ability to alter the execution of the Action. Supplying an empty list will remove all existing managers.
- **run_monitors** (*Optional[Sequence[str]]*) – A list of Globus Auth URNs which will have the ability to view the execution of the Action. Supplying an empty list will remove all existing monitors.
- **kwargs** – Any additional kwargs passed into this method are passed onto the Globus BaseClient. If there exists an "authorizer" keyword argument, that gets used to run the Flow operation. Otherwise, the authorizer_callback defined for the FlowsClient will be used.
- **tags** (*Optional[Sequence[str]]*) – A list of tags to apply to the Run.
- **label** (*Optional[str]*) – A label to apply to the Run.

Return type
globus_sdk.GlobusHTTPResponse

update_runs(*run_ids*, *add_tags=None*, *remove_tags=None*, *set_tags=None*, *add_run_managers=None*, *remove_run_managers=None*, *set_run_managers=None*, *add_run_monitors=None*, *remove_run_monitors=None*, *set_run_monitors=None*, *status=None*, ***kwargs*)

Update a Flow Action.

Parameters

- **run_ids** (*Iterable[str]*) – A list of Run ID's to query.

- **set_tags** (*Optional[Iterable[str]]*) – A list of tags to set on the specified Run ID's.

  If the list is empty, all tags will be deleted from the specified Run ID's.

  Note: The `set_tags`, `add_tags`, and `remove_tags` arguments are mutually exclusive.

- **add_tags** (*Optional[Iterable[str]]*) – A list of tags to add to each of the specified Run ID's.

  Note: The `set_tags`, `add_tags`, and `remove_tags` arguments are mutually exclusive.

- **remove_tags** (*Optional[Iterable[str]]*) – A list of tags to remove from each of the specified Run ID's.

  Note: The `set_tags`, `add_tags`, and `remove_tags` arguments are mutually exclusive.

- **set_run_managers** (*Optional[Iterable[str]]*) – A list of Globus Auth URN's to set on the specified Run ID's.

  If the list is empty, all Run managers will be deleted from the specified Run ID's.

  Note: The `set_run_managers`, `add_run_managers`, and `remove_run_managers` arguments are mutually exclusive.

- **add_run_managers** (*Optional[Iterable[str]]*) – A list of Globus Auth URN's to add to each of the specified Run ID's.

  Note: The `set_run_managers`, `add_run_managers`, and `remove_run_managers` arguments are mutually exclusive.

- **remove_run_managers** (*Optional[Iterable[str]]*) – A list of Globus Auth URN's to remove from each of the specified Run ID's.

  Note: The `set_run_managers`, `add_run_managers`, and `remove_run_managers` arguments are mutually exclusive.

- **set_run_monitors** (*Optional[Iterable[str]]*) – A list of Globus Auth URN's to set on the specified Run ID's.

  If the list is empty, all Run monitors will be deleted from the specified Run ID's.

  Note: The `set_run_monitors`, `add_run_monitors`, and `remove_run_monitors` arguments are mutually exclusive.

- **add_run_monitors** (*Optional[Iterable[str]]*) – A list of Globus Auth URN's to add to each of the specified Run ID's.

Note: The `set_run_monitors`, `add_run_monitors`, and `remove_run_monitors` arguments are mutually exclusive.

- **remove_run_monitors** (*Optional[Iterable[str]]*) – A list of Globus Auth URN's to remove from each of the specified Run ID's.

Note: The `set_run_monitors`, `add_run_monitors`, and `remove_run_monitors` arguments are mutually exclusive.

- **status** (*Optional[str]*) – A status to set for all specified Run ID's.
- **kwargs** – Any additional keyword arguments passed to this method are passed to the Globus BaseClient.

  If an "authorizer" keyword argument is passed, it will be used to authorize the Flow operation. Otherwise, the authorizer_callback defined for the FlowsClient will be used.

Raises

ValueError – If more than one mutually-exclusive argument is provided. For example, if `set_tags` and `add_tags` are both specified, or if `add_run_managers` and `remove_run_managers` are both specified.

Return type

globus_sdk.GlobusHTTPResponse

flow_action_log(*flow_id*, *flow_scope*, *flow_action_id*, *limit=10*, *reverse_order=False*, *marker=None*, *per_page=None*, ***kwargs*)

Retrieve an Action's execution log history for an Action that was launched by a specific Flow.

Parameters

- **flow_id** (*str*) – The UUID identifying the Flow which launched the Action
- **flow_scope** (*str*) – The scope associated with the Flow `flow_id`. If not provided, the SDK will attempt to perform an introspection on the Flow to determine its scope automatically
- **flow_action_id** (*str*) – The ID specifying which Action's history to query
- **limit** (*int*) – An integer specifying the maximum number of records for the Action's execution history to return.
- **reverse_order** (*bool*) – An indicator for whether to retrieve the records in reverse-chronological order.
- **marker** (*Optional[str]*) – A pagination_token indicating the page of results to return and how many entries to return. This is created by the Flows service and returned by operations that support pagination.
- **per_page** (*Optional[int]*) – The number of results to return per page. If supplied a pagination_token, this parameter has no effect.
- **kwargs** – Any additional kwargs passed into this method are passed onto the Globus BaseClient. If there exists an "authorizer" keyword argument, that gets used to run the Flow operation. Otherwise, the authorizer_callback defined for the FlowsClient will be used.

Return type

globus_sdk.GlobusHTTPResponse

classmethod new_client(*client_id*, *authorizer_callback*, *authorizer*, *base_url=None*, *http_timeout=10*)

Classmethod to simplify creating an FlowsClient. Use this method when attemping to create a FlowsClient with pre-existing credentials or authorizers. This method is useful when creating a FlowClient for interacting with a Flow without wanting to launch an interactive login process.

Parameters

- **client_id** (*str*) – The client_id to associate with this FlowsClient.
- **authorizer_callback** (*Callable[[...], Union[globus_sdk.AccessTokenAuthorizer, globus_sdk.RefreshTokenAuthorizer, globus_sdk.ClientCredentialsAuthorizer]]*) – A callable which is capable of returning an authorizer for a particular Flow. The callback should accept three keyword-args: flow_url, flow_scope, client_id. Using some, all, or none of these args, the callback should return a GlobusAuthorizer which provides access to the targetted Flow.
- **authorizer** (*Optional[globus_sdk.authorizers.GlobusAuthorizer]*) – The authorizer to use for validating requests to the Flows service. This authorizer is used when interacting with the Flow's service, it is not used for interactive with a particular flow. Therefore, this authorizer should grant consent to the MANAGE_FLOWS_SCOPE. For interacting with a particular flow, set the authorizer_callback parameter.
- **base_url** (*Optional[str]*) – The url at which the target Action Provider is located.
- **http_timeout** (*int*) – The amount of time to wait for connections to the Action Provider to be made.

Return type

_FlowsClient

Examples

```
>>> def cli_authorizer_callback(**kwargs):
        flow_url = kwargs["flow_url"]
        flow_scope = kwargs["flow_scope"]
        client_id = kwargs["client_id"]
        return get_cli_authorizer(flow_url, flow_scope, client_id)
>>> action_url = "https://actions.globus.org/hello_world"
>>> client_id = "00000000-0000-0000-0000-000000000000"
>>> auth = ...
>>> fc = FlowsClient.new_client(client_id, cli_authorizer_callback, auth)
>>> print(fc.list_flows())
```

10.3 Helpers

If you're running the SDK locally in a secure location, you can use the provided convenience functions, `create_action_client` and `create_flows_client`, to create authenticated `ActionClient` and `FlowsClient` instances. These helpers share functionality with the `globus-automate` CLI to handle loading and storing `Globus Auth` tokens on the local filesystem. They will also trigger interactive logins when additional consents are required to interact with `Actions` or `Flows`.

`globus_automate_client.`**create_action_client**(*action_url*, *action_scope=None*, *client_id='e6c75d97-532a-4c88-b031-8584a319fa3e'*)

A helper function to handle creating a properly authenticated `ActionClient` which can operate against *Globus Action Provider Interface* compliant Action Providers. This helper will create an `ActionClient` by searching

for and storing tokens on the local filesystem, potentially triggering a log-in flow if the requested tokens are not found locally.

Given the `action_url` for a specific ActionProvider, this function will attempt to create a valid `ActionClient` for interacting with that ActionProvider. If the `action_scope` is not provided, this function will attempt to discover the `action_scope` by querying the target Action Provider's introspection endpoint. If the Action Provider is not configured to allow public, unauthenticated access to its introspection endpoint, the `action_scope` will be non-discoverable and authentication will fail.

With the `action_scope` available, the function will search for a valid token in the local filesystem cache. In the event that tokens for the scope cannot be loaded, an interactive login will be triggered. Once tokens have been loaded, an Authorizer is created and used to instantiate the `ActionClient` which can be used for operations against that Action Provider.

Parameters

- **action_url** (*str*) – The URL address at which the target Action Provider exists
- **action_scope** (*Optional[str]*) – The target Action Provider's Globus Auth Scope used for authenticating access to it
- **client_id** (*str*) – The ID for the Native App Auth Client which will be triggering the login flow for this ActionClient

Return type
ActionClient

Examples

```
>>> from globus_automate_client import create_action_client
>>> # Create an ActionClient for the HelloWorld Action
>>> ac = create_action_client("https://actions.globus.org/hello_world")
>>> # Run an Action and check its results
>>> resp = ac.run({"echo_string": "Hello from SDK"})
>>> assert resp.data["status"] == "SUCCEEDED"
```

globus_automate_client.**create_flows_client**(*client_id='e6c75d97-532a-4c88-b031-8584a319fa3e'*, *base_url='https://flows.globus.org'*, *scope='https://auth.globus.org/scopes/eec9b274-0c81-4334-bdc2-54e90e689b9a/manage_flows'*, ***, *authorizer=None*, *authorizer_callback=<function cli_authorizer_callback>*, *http_timeout=10*)

A helper function to handle creating a properly authenticated `FlowsClient` which can operate against the Globus Automate Flows service. This function will attempt to load tokens for the `MANAGE_FLOWS_SCOPE``from the local filesystem, triggering a log-in if the requested tokens are not found locally. Once tokens have been loaded, an Authorizer is created and used to instantiate the ``FlowsClient`. Attempts to interact with a specific Flow will similarly search for valid tokens in the local cache, triggering an interactive log-in if they cannot be found.

Parameters

- **scope** (*str*) – The Globus Auth scope to which the FlowsClient should be created with consents to
- **client_id** (*str*) – The Globus ID to associate with this instance of the FlowsClient
- **base_url** (*str*) – The URL at which the Globus Automate Flows service is located
- **authorizer** (*Optional[globus_sdk.authorizers.GlobusAuthorizer]*) – An authorizer providing access to the Flows service. If not provided, it will be created using the

authorizer_callback

- **authorizer_callback** (*Callable*) – A callback used to dynamically return GlobusAuthorizers. If not provided, the Globus Automate CLI callback will be used which triggers interactive logins and stores tokens locally
- **http_timeout** (*int*) – Close any requests taking longer than this parameter's value

Return type
FlowsClient

Examples

```
>>> from globus_automate_client import create_flows_client
>>> # Create an authenticated FlowsClient that can run operations against the Flows
>>> # service
>>> fc = create_flows_client()
>>> # Get a listing of runnable, deployed flows
>>> available_flows = fc.list_flows(["runnable_by"])
>>> for flow in available_flows.data["flows"]:
>>>     print(flow)
```

10.4 Examples

10.4.1 Non-interactive Operations

There are plenty of scenarios in which the `create_action_client` and `create_flows_client` helper functions should *NOT* be used, included in those are:

- The SDK is being run on a platform without access to the local filesystem (or it is preferable not to write to the local filesystem).
- You do not want to (or cannot) trigger interactive logins during execution.
- The SDK is being used as part of another service or in a pipeline

In this case, you'll need to create the clients using the `new_client` classmethod attached to each client. When creating an ActionClient, you'll need to create and supply the appropriate GlobusAuthorizer for the action.

When creating a FlowsClient, you'll need to create a GlobusAuthorizer with access to the MANAGE_FLOWS_SCOPE. This authorizer will be used to authenticate interactions against the `Flows` API, such as creating a new Flow, updating an existing Flow, or deleting an old Flow. Additionally, you'll need to create a callback that accepts three keyword-arguments, `flow_url`, `flow_scope` and `client_id` and returns a GlobusAuthorizer that will be used to provide access to specific Flows. This authorizer allows the SDK to run operations against the Flow, such as running a Flow and checking an execution's status.

The example below shows an example defining the callback used to create a GlobusAuthorizer by pulling tokens from environment variables.

```
import os

from globus_sdk import AccessTokenAuthorizer

from globus_automate_client import FlowsClient
from globus_automate_client.cli.auth import CLIENT_ID
from globus_automate_client.flows_client import AllowedAuthorizersType
```

(continues on next page)

(continued from previous page)

```
def authorizer_retriever(
    flow_url: str, flow_scope: str, client_id: str
) -> AllowedAuthorizersType:
    """
    This callback will be called when attempting to interact with a
    specific Flow. The callback will receive the Flow url, Flow scope, and
    client_id and can choose to use some, all or none of the kwargs. This is
    expected to return an Authorizer which can be used to make authenticated
    calls to the Flow.

    The method used to acquire valid credentials is up to the user. Here, we
    naively create an Authorizer using the same token everytime.
    """
    flow_token = os.environ.get("MY_ACCESS_TOKEN", "")
    return AccessTokenAuthorizer(flow_token)


# Create an AccessTokenAuthorizer using a token that has consents to the
# MANAGE_FLOWS_SCOPE. This lets the FlowsClient perform operations against the
# Flow's service i.e. create flow, update a flow, delete a flow
flows_service_token = os.environ.get("MANAGE_FLOWS_SCOPED_TOKEN", "")
flows_service_authorizer = AccessTokenAuthorizer(flows_service_token)

fc = FlowsClient.new_client(
    client_id=CLIENT_ID,
    authorizer=flows_service_authorizer,
    authorizer_callback=authorizer_retriever,
)

my_flows = fc.list_flows()
print(my_flows)

# When running a specific Flow, the authorizer_retriever callback is called
# internally to make the authenticated call to the Flow
running_flow = fc.run_flow(
    "1e6b4406-ee3d-4bc5-9198-74128e108111", None, {"echo_string": "hey"}
)
print(running_flow)

# It's possible to create an Authorizer and pass it as a kwarg to the flow
# operation. This usage will not use the authorizer_callback:
running_flow_2 = fc.run_flow(
    "1e6b4406-ee3d-4bc5-9198-74128e108111",
    None,
    {"echo_string": "hey"},
    authorizer=AccessTokenAuthorizer("..."),
)
print(running_flow_2)
```

CHAPTER

ELEVEN

EXAMPLE FLOWS

For examples of Flows, input schemas, and inputs see the examples in the Globus Automate Client repository.

CHAPTER

TWELVE

CHANGELOG

12.1 Unreleased changes

Unreleased changes are documented in files in the changelog.d directory.

12.2 0.16.1.post1 — 2022-07-15

12.2.1 Documentation

- [sc-16711] Document how to automatically run a flow based on filesystem change events.

12.3 0.16.1 — 2022-06-23

12.3.1 Bugfixes

- Fix a bug in the flow administrator/starter/runner CLI alias validation which prevents successfully running the `flow update` subcommand.

12.4 0.16.0 — 2022-06-23

12.4.1 Changes

- [sc-13892] Standardize flow viewer, starter, and administrator arguments – and their aliases – throughout the CLI and API.

  Although this change fixes bugs that prevented documented CLI options from working, it also introduces breaking changes:

  - `--flow-viewer`, `--flow-starter`, and `--flow-administrator` are the canonical CLI options.
  - The CLI option aliases will continue to work, but the alias behavior has changed:
    - If an alias is used, a warning is written to STDERR.
    - If the canonical option name is combined with one of its aliases, an error will be written to STDERR and the Automate client will exit. The exit code will be 1.

    * If more than one of the acceptable aliases is used (like `--starter` and `--runnable-by`), an error will be written to STDERR and the Automate client will exit. The exit code will be 1.

In addition, there are breaking changes in the API:

- `flow_viewers`, `flow_starters`, and `flow_administrators` are the canonical API argument names.
- The API argument aliases will continue to work, but the alias behavior has changed:
    * If an alias is used, a Python warning will be issued. Applications can control the warning behavior using Python's `warnings` module.
    * If the canonical option name is combined with one of its aliases, a `ValueError` will be raised.
    * If more than one of the acceptable aliases is used (like `starters` and `runnable_by`), a `ValueError` will be raised.

12.4.2 Bugfixes

- Update pyjwt to version 2.4.0 to address CVE-2022-29217.

12.5 0.15.2 — 2022-06-06

- [sc-16137] Fix flows run dry-run URL path

12.6 0.15.1 — 2022-05-09

12.6.1 Bugfixes

- Fix a missing dependency when running on Python 3.10.

  This was fixed by adding typing-extensions as an explicit dependency. Note that pip may need to be upgraded due to changes in its dependency resolver.

12.7 0.15.0.post1 — 2022-05-09

NOTE:

This release contains no changes from 0.15.0. It adds text to the changelog regarding an update to the rich package dependency.

12.7.1 Documentation

- [sc-13642] Provide two examples of looping in flow definitions:
    - How to loop a set number of times
    - How to perform batch processing over an unknown quantity of items

12.7.2 Development

- Update click to version 8.0.4. This resolves a security issue.
- Update typer to version 0.4.1.
- Update scriv to version 0.14.0. scriv is a development dependency.
- Update rich to 0.12.3.

  This resolves a dependency conflict between the Globus CLI and the Globus Automate CLI. Both command line clients can now be installed in the same environment.
- Temporarily remove typer-cli as a listed development dependency. It is still needed when generating the CLI documentation.
- Add safety as a test environment for local and CI testing.
- Test against Python 3.10 in CI.
- [sc-14485] Disable SSL verification when interacting with a local development server.

12.8 0.15.0 — 2022-04-29

12.8.1 Documentation

- [sc-13642] Provide two examples of looping in flow definitions:
  - How to loop a set number of times
  - How to perform batch processing over an unknown quantity of items

12.8.2 Development

- Update click to version 8.0.4. This resolves a security issue.
- Update typer to version 0.4.1.
- Update scriv to version 0.14.0. scriv is a development dependency.
- Temporarily remove typer-cli as a listed development dependency. It is still needed when generating the CLI documentation.
- Add safety as a test environment for local and CI testing.
- Test against Python 3.10 in CI.
- [sc-14485] Disable SSL verification when interacting with a local development server.

12.9 0.14.1 — 2022-04-13

12.9.1 Bugfixes

- Changed text on improper token cache file condition so that it doesn't reference the Timer CLI

12.10 0.14.0 — 2022-03-25

12.10.1 Features

- [sc-13426] Support setting tags when using the `flow run` subcommand.
- Support batch updates of one or more Runs.
- Support updating tags and labels using the `flow run-update` subcommand.
- Support erasing the list of Run managers and Run monitors using the `flow run-update` subcommand. This can be done by specifying an empty string for the value of the `--run-manager` and `--run-monitor` options.

12.10.2 Bugfixes

- [sc-13664] Fix tabular `run-list` output.
- [sc-14109] Mark the `run-status` subcommand's `--flow-id` option as a mandatory UUID.
- [sc-14127] Prevent a validation error that occurs when an input schema is not provided to the `flow deploy` subcommand.

12.11 0.13.1 — 2022-03-02

12.11.1 Bugfixes

- Output login prompts to STDERR. This protects serialized output to STDOUT so it can be piped to tools like *jq*.

12.11.2 Documentation

- Documentation and examples for the `globus-collection` input schema format.

12.12 0.13.0 — 2022-02-11

12.12.1 Documentation

- Add the "`notify_on_*`" parameters to the transfer action provider JSON example.
- The description of the Action polling policy has been updated and a discussion of how caching of token validation checks may impact users who invalidate their tokens has been added.
- Adds an input schema for the example single-transfer Flow definition.

- Add documentation for *globus-collection-id* and *globus-collection-path* formats

12.13 0.13.0b2 — 2021-12-09

12.13.1 Bugfixes

- Fix a `KeyError` crash that occurs when enabling verbose output using the `-v` argument. (#111)
- Fix a `ValueError` crash that occurs when displaying a flow. (#110)

12.14 0.13.0b1 — 2021-12-09

12.14.1 Features

- Upgrade to Globus SDK v3.

12.14.2 Bugfixes

- Fixes a bug in the SDK that prevented Flow updates from removing all flow_administrators, flow_viewers, and flow_starters. This bug also prevented updates from setting text fields to empty strings.
- Fix a bug that could allow the Flows authorizer to be lost if an exception was raised. (Authorizer swaps are now handled using a context manager.)
- Support strings (and tuples/sets containing strings) as argument values when running, deploying, or updating an action or a flow and specifying a keyword argument alias like `visible_to` or `runnable_by`.

12.14.3 Other

- Add code linting, documentation build testing, and a bunch of unit tests.
- Add GitHub Actions to run on push and pull requests.
- Add a pre-commit configuration file to increase overall code quality.

12.15 0.12.3 — 2021-11-10

12.15.1 Bugfixes

- Fix a bug that prevented the Flows client from properly validating flow definition states in lists.
- Prevent empty values from appearing in query parameters or JSON submissions.
- Fix a bug that prevented the input schema of an existing Flow from being set to an all-permissive JSON schema.
- Fix a bug that prevented a custom authorizer from being used if attempting to list all runs of a specific flow without specifying the flow ID.
- Fix a bug that introduced duplicate forward slashes in some API URL's.

12.15.2 Documentation

- Add a CHANGELOG and include it in the documentation.
- Use scriv for CHANGELOG management.
- Added documentation for the new Action Providers for: - Make a directory via Globus Transfer (mkdir) - Get collection information via Globus Transfer (collection_info)
- Added documentation for new feature of the list directory Action Provider to get information only about the path provided as input
- Added documentation related to use of built-in functions in expressions. Documented the new functions `pathsplit`, `is_present` and `getattr`.

12.16 0.12.2 — 2021-10-19

12.16.1 Features

- The output of globus-automate flow list is modified to ensure that the Flow ID is always visible. The new output search is similar to the output of `globus endpoint search`.
- The output when watching the results of a `globus-automate flow run` now defaults to table view.

12.16.2 Bugfixes

- Fixes an infinite loop when watching the output of `flow action-log/flow run-log` with the `--reverse` flag.
- Fixes the limit minimum level from 0 to 1 when doing `flow action-log/flow run-log` to prevent server errors.
- Fixes a bug where an unsafe indexing operation was being made during `flow action-list/flow run-list`

12.16.3 Documentation

- CLI documentation is updated to more precisely reflect that `--label` is a required property when running a Flow.

12.17 0.12.1 — 2021-09-14

12.17.1 Features

- CLI commands which output lists of data now display a subset of the data fields as a table. For access to the full data or to access data in JSON or YAML format, the `-f json | yaml` option may be used. The tabular output is on the following commands: - `globus-automate flow list` - `globus-automate flow action-list ...` - `globus-automate flow action-enumerate ...` - `globus-automate flow action-log ...`
- File inputs to CLI commands may now be in either JSON or YAML formatting without the need to specify the input file format.

12.17.2 Bugfixes

- Fixed an error where the output of the `globus-automate flow update` command displayed unformatted JSON

12.17.3 Documentation

- Added text explaining that the Fail state is a supported state type and can be used in Flows. A simple example using the Fail state is included in the examples directory for the repository.

12.18 0.12.0 — 2021-08-16

12.18.1 Features

- CLI and SDK support for updating user roles on new and existing Runs
- Wherever identities are referenced on the CLI we now support supplying Globus Auth usernames instead.
- Updates to CLI and SDK arguments to more closely reflect RBAC updates in the Flows service.

12.18.2 Bugfixes

- The Run enumeration CLI and SDK methods would attempt to use the Flow manage authorizer to authenticate its calls. This method has been updated to instead look up or create an authorizer for the RUN_STATUS scope

12.18.3 Documentation

- The RBAC system for the Flows service has been updated to follow a subset model rather than the previously existing separate permissions model. The documentation has been updated with a description of the new behavior.

12.19 0.11.5 — 2021-06-17

12.19.1 Features

- Adds SDK and CLI support for dry running a Flow deploy or Flow run
- Adds SDK + CLI commands for enumerating Actions and sorting/filtering through results
- Adds a CLI command to retrieve a single Flow definition and its metadata: `globus-automate flow get <id>`
- Expands the use of the `create_flows_client` function to allow specifying an authorizer, an authorizer callback, and a http_timeout.

12.19.2 Bugfixes

- Fixes a regression where Flow deploy results via the CLI were unformatted
- Adds license to output of `pip show globus-automate-client`

12.19.3 Documentation

- Fixes an issue where `FlowsClient` and `ActionClient` auto-generated docs were not getting generated
- Adds references to exemplar Flows and their inputs
- Adds input examples to Action Provider reference page
- Adds a hosted CLI reference

12.20 0.11.4 — 2021-05-10

12.20.1 Features

- The CLI and SDK now allow Subscription IDs to be associated with Flows

12.20.2 Bugfixes

- The Flow List CLI and SDK operations were sending malformed query arguments to the API, which produced incorrect results when trying to filter based on role. This release corrects the behavior.

12.21 0.11.3 — 2021-05-04

12.21.1 Bugfixes

- Reformats verbose output to make the separation between request information and request results more obvious
- Verbose output writes output to `stderr` to allow output to be parsed as `JSON`
- Empty query arguments are not sent as part of the Flows API request

12.21.2 Documentation

- Typo fixes

12.22 0.11.1 — 2021-04-08

12.22.1 Features

- `flow display` can now visualize local Flow definitions and deployed Flows.

12.22.2 Bugfixes

- Fixes an issue where the Globus Auth login link was being rendered as a non-clickable link.
- Fixes an issue where the prompt for inputting the Globus Auth auth code was disappearing.

12.22.3 Documentation

- Adds explanation and examples for how to use `manage_by` and `monitor_by` values on Actions and Flow runs to delegate access to other identities.
- Clarifies the expected format for provided identities.
- Explicitly adds `manage_by` and `monitor_by` as parameters to the `FlowsClient.run` method.

12.23 0.11.0 — 2021-03-29

12.23.1 Features

- Export the `validate_flow_definition` function which can be used to perform a local JSONSchema based validation of a Flow definition.
- Using `create_flows_client` no longer requires the use of a `CLIENT_ID`.
- The `action run`, `action status`, `flow run`, `flow status`, and `flow log` CLI commands implement a new `--watch` flag which lets you stream an Action's status updates.
- CLI and SDK level support for filtering and ordering Flow Listing and Flow Action Enumerations endpoints [preview].
- New CLI commands to facilitate the following `Globus Auth` operations: - `session whoami` - determine the caller's user information as it exists in Auth - `session logout` - remove locally cached auth state - `session revoke` - invalidate local tokens and remove locally cached auth state.

12.23.2 Documentation

- Various typo fixes.

12.24 0.10.7 — 2021-02-11

12.24.1 Features

- Improved error handling on CLI operations so that users receive formatted output instead of `GlobusAPIError` tracebacks.
- Added CLI and SDK level support for using `label`s to launch Flows and Actions.

12.24.2 Documentation

- Removes references to `ActionScope` from example Flow definitions because the Flows service handles the scope lookups.

12.24.3 Bugfixes

- The Flows CLI interface would attempt to load empty arguments, resulting in `NoneType` errors. Empty arguments are now ignored.
- When using the CLI with the `--verbose` flag, the results of the verbosity are printed to `stderr`, allowing the commands outputs to still be parsed by other tools, such as `jq`.
- Fixes a `NameTooLong` exception that was thrown when the CLI attempted to parse long JSON strings as filenames.

12.25 0.10.6 — 2021-01-27

12.25.1 Features

- Adds support for YAML formatted input when defining Flows, input schemas, and inputs via the CLI.

12.25.2 Documentation

- Improves documentation around manually creating authorizers and how to use them to create `ActionClients` and `FlowsClient`: https://globus-automate-client.readthedocs.io/en/latest/python_sdk.html#sdk-the-hard-way
- Adds examples for Flow definitions as YAML: https://github.com/globus/globus-automate-client/tree/main/examples/flows/hello-world-yaml

12.26 0.10.5 — 2020-12-11

12.26.1 Features

- Removes custom SSH session detection in favor of using fair-research native-login's SSH session detection
- Adds Flows pagination support to CLI and SDK layers
- Fully decouples the SDK from the CLI. SDK users can now opt to supply their own authorizers for Flow operations, either as a kwargs to the operation or as a callback to the FlowsClient which should be used to lookup the appropriate authorizer.

12.26.2 Documentation

- Fixes typos in Flow's documentation where Private_Parameters were incorrectly referenced as Private_Properties
- Publishes a new example Flow for performing a multi-step Transfer & delete, along with error checking

12.27 0.10.4 — 2020-10-01

12.27.1 Features

- Added support for deleting messages off a Globus Queue to the CLI and SDK
- Adds example action bodies to the repository for running an action on the new Search Delete Action Provider
- Updated docs and example action bodies for running an action on the Set Permissions Action Provider
- Updates the schema validation for the Pass State to make Parameters and InputPath optional.

12.27.2 Bugfixes

- Corrected an issue in CLI option validation where "public" and "all_authenticated_users" were not being accepted
- Corrected an issue where the SDK's ActionClient was setting monitor_by and manage_by to None by default, thus failing Action Provider schema validation.

12.28 0.10 — 2020-08-24

This release is the first based on the public globus-automate-client repository. Compared to previous PyPi releases, this release contains:

- A more complete set of documentation which is also published to readthedocs
- A set of examples under the examples directory
- Client side validation of flow definitions based on a jsonschema. This is somewhat experimental at this point, and feedback is welcome on experience both with the accuracy and the helpfulness of the reported errors. Validation is turned on by default when deploying or linting a flow, but can be turned off with the SDK parameter `validate_definition` and the CLI `--validate/no-validate` flags.

PYTHON MODULE INDEX

INDEX

S

U