Craft AI MLOps platform Python SDK

Release 0.61.1rc1

Craft AI

Mar 11, 2025

Contents

CHAPTER

ONE

craft_ai_sdk package

1.1 craft_ai_sdk.exceptions module

exception craft_ai_sdk.exceptions.**SdkException**(*message*, *status_code=None*, *name=None*, *request_id=None*, *additional_data=None*)

Bases: Exception

General exception while using this package

1.2 craft_ai_sdk.sdk module

See section *CraftAiSdk class* for more details.

CHAPTER

TWO

`CraftAiSdk` class

class craft_ai_sdk.**CraftAiSdk**(*sdk_token=None*, *environment_url=None*, *control_url=None*, *verbose_log=None*, *warn_on_metric_outside_of_step=True*)

Main class to instantiate

base_environment_url

Base URL to CraftAI Environment.

Type
str

base_environment_api_url

Base URL to CraftAI Environment API.

Type
str

base_control_url

Base URL to CraftAI authorization server.

Type
str

base_control_api_url

Base URL to CraftAI authorization server API.

Type
str

verbose_log

If True, information during method execution will be printed.

Type
bool

warn_on_metric_outside_of_step

If True, a warning will be printed when a metric is added outside of a step.

Type
bool

__init__(*sdk_token=None*, *environment_url=None*, *control_url=None*, *verbose_log=None*, *warn_on_metric_outside_of_step=True*)

Inits CraftAiSdk.

Parameters

- **sdk_token** (str, optional) – SDK token. You can retrieve it from the website. Defaults to CRAFT_AI_SDK_TOKEN environment variable.
- **environment_url** (str, optional) – URL to CraftAI environment. Defaults to CRAFT_AI_ENVIRONMENT_URL environment variable.

- **control_url** (str, optional) – URL to CraftAI authorization server. You probably don't need to set it. Defaults to CRAFT_AI_CONTROL_URL environment variable, or https://mlops-platform.craft.ai.
- **verbose_log** (bool, optional) – If True, information during method execution will be printed. Defaults to True if the environment variable SDK_VERBOSE_LOG is set to true; False if it is set to false; else, defaults to True in interactive mode; False otherwise.
- **warn_on_metric_outside_of_step** (bool, optional) – If True, a warning will be raised when a metric is added outside of a step. Defaults to True.

Raises

- **ValueError** – if the sdk_token or environment_url is not defined and
- **the corresponding environment variable is not set.** –

2.1 Step

CraftAiSdk.**create_step**(*step_name*, *function_path=None*, *function_name=None*, *description=None*, *container_config=None*, *inputs=None*, *outputs=None*, *wait_for_completion=True*, *timeout_s=None*)

Create pipeline step from a function located on a remote repository or locally.

Use *CREATION_PARAMETER_VALUE* to explicitly set a value to null or fall back on project information. You can also use container_config.included_folders to specify the files and folders required for the step execution. This is useful if your repository contains large files that are not required for the step execution, such as documentation or test files. Indeed there is a maximum limit of 5MB for the total size of the content specified with included_folders.

Parameters

- **step_name** (str) – Step name.
- **function_path** (str, optional) – Path to the file that contains the function. This parameter is required if parameter "dockerfile_path" is not specified.
- **function_name** (str, optional) – Name of the function in that file. This parameter is required if parameter "dockerfile_path" is not specified.
- **description** (str, optional) – Description. Defaults to None.
- **container_config** (dict[str, str], optional) – Some step configuration, with the following optional keys:
  - "language" (str): Language and version used for the step. Defaults to falling back on project information. The accepted formats are "python:3.X-slim", where "3.X" is a supported version of Python, and "python-cuda:3.X-Y.Z" for GPU environments, where "Y.Z" is a supported version of CUDA. The list of supported versions is available on the official documentation website at https://mlops-platform-documentation.craft.ai.
  - "repository_url" (str): Remote repository url. Defaults to falling back on project information.
  - "repository_branch" (str): Branch name. Defaults to falling back on project information.
  - "repository_deploy_key" (str): Private SSH key of the repository. Defaults to falling back on project information, can be set to null. The

key should retain the header/footer beacon with : "BEGIN/END RSA PRIVATE KEY".

  - `"requirements_path"` (`str`): Path to the requirements.txt file. Environment variables created through *`create_or_update_environment_variable()`* can be used in requirements.txt, as in `"${ENV_VAR}"`. Defaults to falling back on project information, can be set to null.
  - `"included_folders"` (`list[str]`): List of folders and files in the repository required for the step execution. Defaults to falling back on project information, can be set to null. Total size of included_folders must be less than 5MB.
  - `"system_dependencies"` (`list[str]`): List of system dependencies. Defaults to falling back on project information, can be set to null.
  - `"dockerfile_path"` (`str`): Path to the Dockerfile. This parameter should only be used as a last resort and for advanced use. When specified, the following parameters should be set to null: `"function_path"`, `"function_name"`, `"language"`, `"requirements_path"` and `"system_dependencies"`.
  - `"local_folder` (`str`): Path to local folder where the step is stored.

- **`inputs`** (*list* of instances of *`Input`*) – List of inputs. Each parameter of the step function should be specified as an instance of *`Input`* via this parameter *inputs*. During the execution of the step, the value of the inputs would be passed as function arguments.
- **`outputs`** (*list* of instances of *`Output`*) – List of the step outputs. For the step to have outputs, the function should return a `dict` with the name of the output as keys and the value of the output as values. Each output should be specified as an instance of *`Output`* via this parameter *outputs*.
- **`wait_for_completion`** (`bool`, optional) – Whether to wait for the step to be created. Defaults to `True`.
- **`timeout_s`** (`int`) – Maximum time (in seconds) to wait for the step to be created. Set to None to wait indefinitely. Defaults to None. Only applicable if `wait_for_completion` is `True`.

Returns

Created step represented as a `dict` with the following keys:

- `"parameters"` (`dict`): Information used to create the step with the following keys:
  - `"step_name"` (`str`): Name of the step.
  - `"function_path"` (`str`): Path to the file that contains the function.
  - `"function_name"` (`str`): Name of the function in that file.
  - `"description"` (`str`): Description.
  - `"inputs"` (`list` of `dict`): List of inputs represented as a `dict` with the following keys:
    - `"name"` (`str`): Input name.
    - `"data_type"` (`str`): Input data type.
    - `"is_required"` (`bool`): Whether the input is required.
    - `"default_value"` (`str`): Input default value.

  - "outputs" (list of dict): List of outputs represented as a dict with the following keys:
    * "name" (str): Output name.
    * "data_type" (str): Output data type.
    * "description" (str): Output description.
  - "container_config" (dict[str, str]): Some step configuration, with the following optional keys:
    * "language" (str): Language and version used for the step. The accepted formats are "python:3.X-slim", where "3.X" is a supported version of Python, and "python-cuda:3.X-Y.Z" for GPU environments, where "Y.Z" is a supported version of CUDA. The list of supported versions is available on the official documentation website at https://mlops-platform-documentation.craft.ai.
    * "repository_branch" (str): Branch name.
    * "repository_url" (str): Remote repository url.
    * "included_folders" (list[str]): List of folders and files in the repository required for the step execution.
    * "system_dependencies" (list[str]): List of system dependencies.
    * "dockerfile_path" (str): Path to the Dockerfile.
    * "requirements_path" (str): Path to the requirements.txt file.
- "creation_info" (dict): Information about the step creation:
  - "created_at" (str): The creation date in ISO format.
  - "commit_id" (str): The commit id on which the step was built.
  - "status" (str): The step status, always "ready" when this function returns.
  - "origin" (str): The origin of the step, can be "git_repository" or "local".

Return type
dict

CraftAiSdk.**get_step**(*step_name*, *wait_for_completion=False*, *timeout_s=None*)

Get a single step if it exists.

Parameters

- **step_name** (str) – The name of the step to get.
- **wait_for_completion** (bool, optional) – Whether to wait for the step to be created. Defaults to False.
- **timeout_s** (int) – Maximum time (in seconds) to wait for the step to be created. Set to None to wait indefinitely. Defaults to None. Only applicable if wait_for_completion is True.

Returns

None if the step does not exist; otherwise the step information, with the following keys:

- "parameters" (dict): Information used to create the step with the following keys:
  - "step_name" (str): Name of the step.

  - `"function_path"` (`str`): Path to the file that contains the function.
  - `"function_name"` (`str`): Name of the function in that file.
  - `"description"` (`str`): Description.
  - `"inputs"` (`list` of `dict`): List of inputs represented as a `dict` with the following keys:
    - `"name"` (`str`): Input name.
    - `"data_type"` (`str`): Input data type.
    - `"is_required"` (`bool`): Whether the input is required.
    - `"default_value"` (`str`): Input default value.
  - `"outputs"` (`list` of `dict`): List of outputs represented as a `dict` with the following keys:
    - `"name"` (`str`): Output name.
    - `"data_type"` (`str`): Output data type.
    - `"description"` (`str`): Output description.
  - `"container_config"` (`dict[str, str]`): Some step configuration, with the following optional keys:
    - `"language"` (`str`): Language and version used for the step. The accepted formats are "python:3.X-slim", where "3.X" is a supported version of Python, and "python-cuda:3.X-Y.Z" for GPU environments, where "Y.Z" is a supported version of CUDA. The list of supported versions is available on the official documentation website at https://mlops-platform-documentation.craft.ai.
    - `"repository_url"` (`str`): Remote repository url.
    - `"repository_branch"` (`str`): Branch name.
    - `"included_folders"` (`list[str]`): List of folders and files in the repository required for the step execution.
    - `"system_dependencies"` (`list[str]`): List of system dependencies.
    - `"dockerfile_path"` (`str`): Path to the Dockerfile.
    - `"requirements_path"` (`str`): Path to the requirements.txt file.
- `"creation_info"` (`dict`): Information about the step creation:
  - `"created_at"` (`str`): The creation date in ISO format.
  - `"commit_id"` (`str`): The commit id on which the step was built.
  - `"status"` (`str`): either `"creation_pending"` or `"ready"`.
  - `"origin"` (`str`): The origin of the step, can be `"git_repository"` or `"local"`.

Return type
`dict`

`CraftAiSdk.`**`list_steps`**`()`

Get the list of all steps.

Returns

List of steps represented as `dict` with the following keys:

- `"step_name"` (`str`): Name of the step.
- `"status"` (`str`): either `"creation_pending"` or `"ready"`.

- `"created_at"` (`str`): The creation date in ISO format.

Return type
`list` of `dict`

`CraftAiSdk.`**`delete_step`**(*step_name*, *force_dependents_deletion=False*)

Delete one step.

Parameters

- **`step_name`** (`str`) – Name of the step to delete as defined in the `config.yaml` configuration file.
- **`force_dependents_deletion`** (`bool`, optional) – if True the associated step's dependencies will be deleted too (pipeline, pipeline executions, deployments). Defaults to False.

Returns

The deleted step represented as a `dict` with the following keys:

- `"step_name"` (`str`): Name of the step.

Return type
`dict[str, str]`

`CraftAiSdk.`**`download_step_local_folder`**(*step_name*, *folder*)

Download a step's local folder as a *.tgz* archive.

Only available if the step's `origin` is `"local_folder"`. This archive contains the files that were in the `local_folder` parameter provided during step creation, and that were included based on the step's `container_config` property.

Parameters

- **`step_name`** (`str`) – Name of the step to be downloaded.
- **`folder`** (`str`) – Path to the folder where the file will be saved.

Returns
None

`CraftAiSdk.`**`get_step_logs`**(*step_name*, *from_datetime=None*, *to_datetime=None*, *limit=None*)

Get the logs of a step.

Parameters

- **`step_name`** (`str`) – Name of the step.
- **`from_datetime`** (`datetime.time`, optional) – Datetime from which the logs are collected.
- **`to_datetime`** (`datetime.time`, optional) – Datetime until which the logs are collected.
- **`limit`** (`int`, optional) – Maximum number of logs that are collected.

Returns

List of collected logs represented as dict with the following keys:

- `"timestamp"` (`str`): Timestamp of the log.
- `"message"` (`str`): Log message.

Return type
`list` of `dict`

class craft_ai_sdk.**Input**(*name*, *data_type*, *description=None*, *is_required=None*, *default_value=None*)

Class to specify a step input when creating a step (cf. *CraftAiSdk.create_step()*).

Parameters

- **name** (str) – Name of the input. This corresponds to the name of a parameter of a step function.
- **data_type** (str) – Type of the input: It could be one of "string", "number","boolean", "json", "array" or "file". For convenience, members of the enumeration *INPUT_OUTPUT_TYPES* could be used too.
- **description** (str, optional) – Description. Defaults to None.
- **is_required** (bool, optional) – Specify if an value should be provided at execution time. Defaults to None.
- **default_value** (Any, optional) – A default value for the step input at execution time. The type for *default_value* should match the type specified by *data_type*. Defaults to None.

class craft_ai_sdk.**CREATION_PARAMETER_VALUE**(*enum.Enum*)

Enumeration for creation parameters special values.

FALLBACK_PROJECT = 'FALLBACK_PROJECT'

Special value to indicate that the parameter should be set to the project information value.

NULL = 'NULL'

Special value to indicate that the parameter should be set to *None*.

class craft_ai_sdk.**Output**(*name*, *data_type*, *description=None*)

Class to specify a step output when creating a step (cf. *CraftAiSdk.create_step()*).

Parameters

- **name** (str) – Name of the output. This corresponds to the key of the *dict* returned by the step function.
- **data_type** (str) – Type of the output. It could be one of "string", "number", "boolean", "json", "array" or "file". For convenience, members of the enumeration *INPUT_OUTPUT_TYPES* could be used too.
- **description** (str, optional) – Description. Defaults to None.

class craft_ai_sdk.**INPUT_OUTPUT_TYPES**(*strenum.LowercaseStrEnum*)

Enumeration for Input and Output data types.

ARRAY = 'array'

BOOLEAN = 'boolean'

FILE = 'file'

JSON = 'json'

NUMBER = 'number'

STRING = 'string'

2.2 Pipeline

CraftAiSdk.**create_pipeline**(*pipeline_name*, *function_path=None*, *function_name=None*, *description=None*, *container_config=None*, *inputs=None*, *outputs=None*, *wait_for_completion=True*, *timeout_s=None*, ***kwargs*)

Create a pipeline from a function located on a remote repository or locally.

Use *CREATION_PARAMETER_VALUE* to explicitly set a value to null or fall back on project information. You can also use `container_config.included_folders` to specify the files and folders required for the pipeline execution. This is useful if your repository contains large files that are not required for the pipeline execution, such as documentation or test files. Indeed there is a maximum limit of 5MB for the total size of the content specified with `included_folders`.

Parameters

- **pipeline_name** (`str`) – Name of the pipeline to be created.
- **function_path** (`str`, optional) – Path to the file that contains the function. This parameter is required if parameter "dockerfile_path" is not specified.
- **function_name** (`str`, optional) – Name of the function in that file. This parameter is required if parameter "dockerfile_path" is not specified.
- **description** (`str`, optional) – Description. Defaults to None.
- **container_config** (`dict[str, str]`, optional) – Some pipeline configuration, with the following optional keys:
  - `"language"` (`str`): Language and version used for the pipeline. Defaults to falling back on project information. The accepted formats are "python:3.X-slim", where "3.X" is a supported version of Python, and "python-cuda:3.X-Y.Z" for GPU environments, where "Y.Z" is a supported version of CUDA. The list of supported versions is available on the official documentation website at https://mlops-platform-documentation.craft.ai.
  - `"repository_url"` (`str`): Remote repository url. Defaults to falling back on project information.
  - `"repository_branch"` (`str`): Branch name. Defaults to falling back on project information.
  - `"repository_deploy_key"` (`str`): Private SSH key of the repository. Defaults to falling back on project information, can be set to null. The key should retain the header/footer beacon with : "BEGIN/END RSA PRIVATE KEY".
  - `"requirements_path"` (`str`): Path to the requirements.txt file. Environment variables created through *create_or_update_environment_variable()* can be used in requirements.txt, as in `"${ENV_VAR}"`. Defaults to falling back on project information, can be set to null.
  - `"included_folders"` (`list[str]`): List of folders and files in the repository required for the pipeline execution. Defaults to falling back on project information, can be set to null. Total size of included_folders must be less than 5MB.
  - `"system_dependencies"` (`list[str]`): List of system dependencies. Defaults to falling back on project information, can be set to null.

  - `"dockerfile_path"` (`str`): Path to the Dockerfile. This parameter should only be used as a last resort and for advanced use. When specified, the following parameters should be set to null: `"function_path"`, `"function_name"`, `"language"`, `"requirements_path"` and `"system_dependencies"`.
  - `"local_folder` (`str`): Path to local folder where the pipeline is stored.
- **`inputs`** (*list* of instances of *`Input`*) – List of inputs. Each parameter of the pipeline function should be specified as an instance of *`Input`* via this parameter *inputs*. During the execution of the pipeline, the value of the inputs would be passed as function arguments.
- **`outputs`** (*list* of instances of *`Output`*) – List of the pipeline outputs. For the pipeline to have outputs, the function should return a `dict` with the name of the output as keys and the value of the output as values. Each output should be specified as an instance of *`Output`* via this parameter *outputs*.
- **`wait_for_completion`** (`bool`, optional) – Whether to wait for the pipeline to be created. Defaults to `True`.
- **`timeout_s`** (`int`) – Maximum time (in seconds) to wait for the pipeline to be created. Set to None to wait indefinitely. Defaults to None. Only applicable if `wait_for_completion` is `True`.

Returns

Created pipeline represented as `dict` with the following keys:

- `"parameters"` (`dict`): Information used to create the pipeline with the following keys:
  - `"pipeline_name"` (`str`): Name of the pipeline.
  - `"function_path"` (`str`): Path to the file that contains the function.
  - `"function_name"` (`str`): Name of the function in that file.
  - `"description"` (`str`): Description.
  - `"inputs"` (`list` of `dict`): List of inputs represented as a `dict` with the following keys:
    - `"name"` (`str`): Input name.
    - `"data_type"` (`str`): Input data type.
    - `"is_required"` (`bool`): Whether the input is required.
    - `"default_value"` (`str`): Input default value.
  - `"outputs"` (`list` of `dict`): List of outputs represented as a `dict` with the following keys:
    - `"name"` (`str`): Output name.
    - `"data_type"` (`str`): Output data type.
    - `"description"` (`str`): Output description.
  - `"container_config"` (`dict[str, str]`): Some pipeline configuration, with the following optional keys:
    - `"language"` (`str`): Language and version used for the pipeline. The accepted formats are "python:3.X-slim", where "3.X" is a supported version of Python, and "python-cuda:3.X-Y.Z" for GPU environments, where "Y.Z" is a supported version of CUDA. The list of supported versions is available on the official documentation website at https://mlops-platform-documentation.craft.ai.

        - `"repository_url"` (`str`): Remote repository url.
        - `"repository_branch"` (`str`): Branch name.
        - `"included_folders"` (`list[str]`): List of folders and files in the repository required for the pipeline execution.
        - `"system_dependencies"` (`list[str]`): List of system dependencies.
        - `"dockerfile_path"` (`str`): Path to the Dockerfile.
        - `"requirements_path"` (`str`): Path to the requirements.txt file.
    - `"creation_info"` (`dict`): Information about the pipeline creation:
      - `"created_at"` (`str`): The creation date in ISO format.
      - `"commit_id"` (`str`): The commit id on which the pipeline was built.
      - `"status"` (`str`): either `"creation_pending"` or `"ready"`.
      - `"origin"` (`str`): The origin of the pipeline, can be `"git_repository"` or `"local"`.
      - `"last_execution_id"` (`str`): The id of the last execution of the pipeline.

  Return type
  `dict`

`CraftAiSdk.`**`get_pipeline`**(*pipeline_name*, *wait_for_completion=False*, *timeout_s=None*)

Get a single pipeline if it exists.

Parameters

- **`pipeline_name`** (`str`) – Name of the pipeline to get.
- **`wait_for_completion`** (`bool`, optional) – Whether to wait for the pipeline to be created. Defaults to `False`.
- **`timeout_s`** (`int`) – Maximum time (in seconds) to wait for the pipeline to be created. Set to None to wait indefinitely. Defaults to None. Only applicable if `wait_for_completion` is `True`.

Returns

None if the pipeline does not exist, otherwise pipeline information, with the following keys:

- `"parameters"` (`dict`): Information used to create the pipeline with the following keys:
  - `"pipeline_name"` (`str`): Name of the pipeline.
  - `"function_path"` (`str`): Path to the file that contains the function.
  - `"function_name"` (`str`): Name of the function in that file.
  - `"description"` (`str`): Description.
  - `"inputs"` (`list` of `dict`): List of inputs represented as a `dict` with the following keys:
    - `"name"` (`str`): Input name.
    - `"data_type"` (`str`): Input data type.
    - `"is_required"` (`bool`): Whether the input is required.
    - `"default_value"` (`str`): Input default value.
  - `"outputs"` (`list` of `dict`): List of outputs represented as a `dict` with the following keys:

        * "name" (str): Output name.
        * "data_type" (str): Output data type.
        * "description" (str): Output description.
      - "container_config" (dict[str, str]): Some pipeline configuration, with the following optional keys:
        * "language" (str): Language and version used for the pipeline. The accepted formats are "python:3.X-slim", where "3.X" is a supported version of Python, and "python-cuda:3.X-Y.Z" for GPU environments, where "Y.Z" is a supported version of CUDA. The list of supported versions is available on the official documentation website at https://mlops-platform-documentation.craft.ai.
        * "repository_url" (str): Remote repository url.
        * "repository_branch" (str): Branch name.
        * "included_folders" (list[str]): List of folders and files in the repository required for the pipeline execution.
        * "system_dependencies" (list[str]): List of system dependencies.
        * "dockerfile_path" (str): Path to the Dockerfile.
        * "requirements_path" (str): Path to the requirements.txt file.
    - "creation_info" (dict): Information about the pipeline creation:
      - "created_at" (str): The creation date in ISO format.
      - "created_by" (str): The user who created the pipeline.
      - "commit_id" (str): The commit id on which the pipeline was built.
      - "status" (str): either "creation_pending" or "ready".
      - "origin" (str): The origin of the pipeline, can be "git_repository" or "local".
      - "last_execution_id" (str): The id of the last execution of the pipeline.

CraftAiSdk.**delete_pipeline**(*pipeline_name*, *force_deployments_deletion=False*)

Delete a pipeline identified by its name.

Parameters

- **pipeline_name** (str) – Name of the pipeline.
- **force_deployments_deletion** (bool, optional) – if True the associated endpoints will be deleted too. Defaults to False.

Returns

The deleted pipeline and its associated deleted deployments represented as a dict with the following keys:

- "pipeline" (dict): Deleted pipeline represented as dict with the following keys:
  - "name" (str): Name of the deleted pipeline.
- "deployments" (list): List of deleted deployments represented as dict with the following keys:
  - "name" (str): Name of the deleted deployments.
  - "execution_rule" (str): Execution rule of the deleted deployments.

Return type
dict

CraftAiSdk.**list_pipelines**()

Get the list of all pipelines.

Returns

List of pipelines represented as dict with the following keys:

- "pipeline_name" (str): Name of the pipeline.
- "created_at" (str): Pipeline date of creation.
- "status" (str): Status of the pipeline.

Return type
list of dict

CraftAiSdk.**download_pipeline_local_folder**(*pipeline_name*, *folder*)

Download a pipeline's local folder as a *.tgz* archive.

Only available if the pipeline's origin is "local_folder". This archive contains the files that were in the local_folder parameter provided during pipeline creation, and that were included based on the pipeline's container_config property.

Parameters

- **pipeline_name** (str) – Name of the pipline to be downloaded.
- **folder** (str) – Path to the folder where the file will be saved.

Returns
None

CraftAiSdk.**get_pipeline_logs**(*pipeline_name*, *from_datetime=None*, *to_datetime=None*, *limit=None*)

Get the logs of a pipeline.

Parameters

- **pipeline_name** (str) – Name of the pipeline.
- **from_datetime** (datetime.time, optional) – Datetime from which the logs are collected.
- **to_datetime** (datetime.time, optional) – Datetime until which the logs are collected.
- **limit** (int, optional) – Maximum number of logs that are collected.

Returns

List of collected logs represented as dict with the following keys:

- "timestamp" (str): Timestamp of the log.
- "message" (str): Log message.

Return type
list of dict

2.3 Pipeline Execution

`CraftAiSdk.`**`run_pipeline`**(*pipeline_name*, *inputs=None*, *inputs_mapping=None*, *outputs_mapping=None*, *wait_for_completion=True*)

Run a pipeline.

Parameters

- **`pipeline_name`** (`str`) – Name of an existing pipeline.
- **`inputs`** (`dict`, optional) – Dictionary of inputs to pass to the pipeline with input names as keys and corresponding values as values. For files, the value should be the path to the file or a file content in an instance of io.IOBase. For json, string, number, boolean and array inputs, the size of all values should be less than 0.06MB. Defaults to None.
- **`inputs_mapping`** (`list` of instances of *`InputSource`*) – List of input mappings, to map pipeline inputs to different sources (constant_value, environment_variable_name, datastore_path or is_null). See *`InputSource`* for more details.
- **`outputs_mapping`** (`list` of instances of *`OutputDestination`*) – List of output mappings, to map pipeline outputs to different destinations (is_null or datastore_path). See *`OutputDestination`* for more details.
- **`wait_for_completion`** (`bool`, optional) – Wait for the end of the execution and returns the execution result. Defaults to *True*.

Returns

Pipeline execution outputs represented as a `dict` with the following keys:

- `"execution_id"` (`str`): Name of the pipeline execution.
- `"outputs"` (`dict`): Pipeline execution outputs with output names as keys and corresponding values as values. Note that this key is only returned if `wait_for_completion` is *True*.

Return type

`dict`

`CraftAiSdk.`**`list_pipeline_executions`**(*pipeline_name*)

Get a list of executions for the given pipeline

Parameters

`pipeline_name` (`str`) – Name of an existing pipeline.

Returns

A list of information on the pipeline execution represented as dict with the following keys:

- `"execution_id"` (`str`): Name of the pipeline execution.
- `"status"` (`str`): Status of the pipeline execution.
- `"created_at"` (`str`): Date of creation of the pipeline execution.
- `"created_by"` (`str`): ID of the user who created the pipeline execution. In the case of a pipeline run, this is the user who triggered the run. In the case of an execution via a deployment, this is the user who created the deployment.
- `"end_date"` (`str`): Date of completion of the pipeline execution.
- `"pipeline_name"` (`str`): Name of the pipeline used for the execution.

- `"deployment_name"` (`str`): Name of the deployment used for the execution.

Return type
`list`

`CraftAiSdk.`**`get_pipeline_execution`**(*execution_id*)

Get the status of one pipeline execution identified by its execution_id.

Parameters
`execution_id` (`str`) – Name of the pipeline execution.

Returns

Information on the pipeline execution represented as dict with the following keys:

- `"execution_id"` (`str`): Name of the pipeline execution.
- `"status"` (`str`): Status of the pipeline execution.
- `"created_at"` (`str`): Date of creation of the pipeline
- `"created_by"` (`str`): ID of the user who created the pipeline execution. In the case of a pipeline run, this is the user who triggered the run. In the case of an execution via a deployment, this is the user who created the deployment.
- `"end_date"` (`str`): Date of completion of the pipeline execution.
- `"pipeline_name"` (`str`): Name of the pipeline used for the execution.
- `"deployment_name"` (`str`): Name of the deployment used for the execution.
- `"steps"` (`list` of *obj*): List of the step executions represented as `dict` with the following keys:
  - `"name"` (`str`): Name of the step.
  - `"status"` (`str`): Status of the step.
  - `"start_date"` (`str`): Date of start of the step execution.
  - `"end_date"` (`str`): Date of completion of the step execution.
  - `"commit_id"` (`str`): Id of the commit used to build the step.
  - `"repository_url"` (`str`): Url of the repository used to build the step.
  - `"repository_branch"` (`str`): Branch of the repository used to build the step.
  - `"requirements_path"` (`str`): Path of the requirements.txt file.
  - `"origin"` (`str`): The origin of the step, can be `"git_repository"` or `"local"`.
- `"inputs"` (`list` of `dict`): List of inputs represented as a dict with the following keys:
  - `"pipeline_input_name"` (`str`): Name of the input.
  - `"data_type` (`str`): Data type of the input.
  - `"source` (`str`): Source of type of the input. Can be “environment_variable”, “datastore”, “constant”, “is_null” “endpoint” or “run”.
  - `"endpoint_input_name"` (`str`): Name of the input in the endpoint execution if source is “endpoint”.

  - "constant_value" (str): Value of the constant if source is "constant".
  - "environment_variable_name" (str): Name of the environment variable if source is "environment_variable".
  - "is_null" (bool): True if source is "is_null".
  - "value": Value of the input.
- "outputs" (list of dict): List of outputs represented as a dict with the following keys:
  - "pipeline_output_name" (str): Name of the output.
  - "data_type (str): Data type of the output.
  - "destination (str): Destination of type of the output. Can be "datastore", "is_null" "endpoint" or "run".
  - "endpoint_output_name" (str): Name of the output in the endpoint execution if destination is "endpoint".
  - "is_null" (bool): True if destination is "is_null".
  - "value": Value of the output.

Return type
dict

CraftAiSdk.**get_pipeline_execution_input**(*execution_id*, *input_name*)

Get the input value of an executed pipeline identified by its execution_id.

Parameters

- **execution_id** (str) – ID of the pipeline execution.
- **input_name** (str) – Name of the input.

Returns

Information on the input represented as a dict with the following keys :

- "pipeline_input_name" (str): Name of the input.
- "data_type (str): Data type of the input.
- "source (str): Source of type of the input. Can be "environment_variable", "datastore", "constant", "is_null" "endpoint" or "run".
- "endpoint_input_name" (str): Name of the input in the endpoint execution if source is "endpoint".
- "constant_value" (str): Value of the constant if source is "constant".
- "environment_variable_name" (str): Name of the environment variable if source is "environment_variable".
- "is_null" (bool): True if source is "is_null".
- "value": Value of the input.

Return type
dict

CraftAiSdk.**get_pipeline_execution_output**(*execution_id*, *output_name*)

Get the output value of an executed pipeline identified by its execution_id.

Parameters

- **execution_id** (str) – ID of the pipeline execution.
- **output_name** (str) – Name of the output.

Returns

Information on the output represented as a dict with the following keys :

- `"pipeline_output_name"` (`str`): Name of the output.
- `"data_type` (`str`): Data type of the output.
- `"destination` (`str`): Destination of type of the output. Can be "datastore", "is_null" "endpoint" or "run".
- `"endpoint_output_name"` (`str`): Name of the output in the endpoint execution if destination is "endpoint".
- `"is_null"` (`bool`): True if destination is "is_null".
- `"value"`: Value of the output.

Return type
`dict`

`CraftAiSdk.`**`get_pipeline_execution_logs`**(*execution_id*, *from_datetime=None*, *to_datetime=None*, *limit=None*)

Get the logs of an executed pipeline identified by its name.

Parameters

- **`execution_id`** (`str`) – ID of the pipeline execution.
- **`from_datetime`** (`datetime.time`, optional) – Datetime from which the logs are collected.
- **`to_datetime`** (`datetime.time`, optional) – Datetime until which the logs are collected.
- **`limit`** (`int`, optional) – Maximum number of logs that are collected.

Returns

List of collected logs represented as dict with the following keys:

- `"timestamp"` (`str`): Timestamp of the log.
- `"message"` (`str`): Log message.

Return type
`list` of `dict`

`CraftAiSdk.`**`delete_pipeline_execution`**(*execution_id*)

Delete one pipeline execution identified by its execution_id.

Parameters
`execution_id` (`str`) – Name of the pipeline execution.

Returns

Deleted pipeline execution represented as dict with the following keys:

- `"execution_id"` (`str`): Name of the pipeline execution.

Return type
`dict`

2.4 Deployment

CraftAiSdk.**create_deployment**(*pipeline_name*, *deployment_name*, *execution_rule*, *mode=DEPLOYMENT_MODES.ELASTIC*, *schedule=None*, *endpoint_url_path=None*, *inputs_mapping=None*, *outputs_mapping=None*, *description=None*, *enable_parallel_executions=None*, *max_parallel_executions_per_pod=None*, *ram_request=None*, *gpu_request=None*, *wait_for_completion=True*, *timeout_s=None*)

Create a deployment associated with a given pipeline.

Parameters

- **pipeline_name** (str) – Name of the pipeline to deploy.
- **deployment_name** (str) – Name of the deployment.
- **execution_rule** (str) – Execution rule of the deployment. Can be "endpoint" or "periodic". For convenience, members of the enumeration *DEPLOYMENT_EXECUTION_RULES* can be used. If the execution rule is "periodic", schedule must be provided. If it is "endpoint", see the note below on how to trigger the endpoint.
- **mode** (str) – Mode of the deployment. Can be "elastic" or "low_latency". Defaults to "elastic". For convenience, members of the enumeration *DEPLOYMENT_MODES* can be used. This defines how computing resources are allocated for pipeline executions:
  - "elastic": Each pipeline execution runs in a new isolated container ("pod"), with its own memory (RAM, VRAM, disk). No variables or files are shared between executions, and the pod is destroyed when the execution ends. This mode is simple to use because it automatically uses computing resources for running executions, and each execution starts from an identical blank state. But it takes time to create a new pod at the beginning of each execution (tens of seconds), and computing resources can become saturated when there are more executions.
  - "low_latency": All pipeline executions for the same deployment run in a shared container ("pod") with shared memory. The pod is created when the deployment is created, and deleted when the deployment is deleted. Shared memory means that if one execution modifies a global variable or a file, subsequent executions on the same pod will see the modified value. This mode makes it possible for executions to respond quickly (less than 0.5 seconds of overhead) because the pod is already up and running when an execution starts, and it is possible to preload or cache data. But it requires care in the code because of possible interactions between executions. And it requires care in how computing resources are used, because pods use resources continuously even when there is no ongoing execution, and the number of pods does not automatically adapt to the number of executions. During the lifetime of a deployment, a pod may be re-created by the platform for technical reasons (including if it tries to use more memory than available). This mode is not compatible with steps created with a container_config.dockerfile_path property in *create_step()*.
- **schedule** (str, optional) – Schedule of the deployment. Required and applicable only if execution_rule is "periodic". Must be a valid *cron expression <https://www.npmjs.com/package/croner>*. The deployment will be executed periodically according to this schedule. The

schedule must follow this format: `<minute> <hour> <day of month> <month> <day of week>`. Note that the schedule is in UTC time zone. '*' means all possible values. Here are some examples:

  - `"0 0 * * *"` will execute the deployment every day at midnight.
  - `"0 0 5 * *"` will execute the deployment every 5th day of the month at midnight.

- **`endpoint_url_path`** (`str`, optional) – Applicable only if `execution_rule` is `"endpoint"`. The last part of the URL (address) used to trigger the endpoint deployment. See the note below on how to trigger the endpoint. By default the value of `deployment_name` is used. This can be used to customize the URL independently of `deployment_name`.
- **`inputs_mapping`** (`list` of instances of *`InputSource`*) – List of input mappings, to map pipeline inputs to different sources (such as constant values, endpoint inputs, or environment variables). See *`InputSource`* for more details. For endpoint rules, if an input of the step in the pipeline is not explicitly mapped, it will be automatically mapped to an endpoint input with the same name. For periodic rules, all inputs of the step in the pipeline must be explicitly mapped.
- **`outputs_mapping`** (`list` of instances of *`OutputDestination`*) – List of output mappings, to map pipeline outputs to different destinations. See *`OutputDestination`* for more details. For endpoint rules, if an output of the step in the pipeline is not explicitly mapped, it will be automatically mapped to an endpoint output with the same name. For periodic rules, all outputs of the step in the pipeline must be explicitly mapped.
- **`description`** (`str`, optional) – Description of the deployment.
- **`enable_parallel_executions`** (`bool`, optional) – Whether to run several executions at the same time in the same pod, if `mode` is `"low_latency"`. Not applicable if `mode` is `"elastic"`, where each execution always runs in a new pod. This is disabled by default, which means that for a deployment with `"low_latency"` `mode`, by default only one execution runs at a time on a pod, and other executions are pending while waiting for the running one to finish. Enabling this may be useful for inference batching on a model that takes much memory, so the model is loaded in memory only once and can be used for several inferences at the same time. If this is enabled then global variables, GPU memory and disk files are shared between multiple executions, so you must be mindful of potential race conditions and concurrency issues. For each execution running on a pod, the main Python function is run either as an `asyncio` coroutine with `await` if the function was defined with `async def` (recommended), or in a new thread if the function was defined simply with `def`. Environment variables are updated whenever a new execution starts on the pod. Using some libraries with async/threaded methods in your code may cause logs to be associated with the wrong running execution (logs are associated with executions through Python `contextvars`).
- **`max_parallel_executions_per_pod`** (`int`, optional) – Only applicable if `enable_parallel_executions` is `True`. The maximum number of executions that can run at the same time on a deployment's pod in `"low_latency"` `mode` where `enable_parallel_executions` is `True`: if a greater number of executions are requested at the same time, then only `max_parallel_executions_per_pod` executions will actually be running on the pod, and the other ones will be pending until a running execution finishes. Defaults to 6.
- **`ram_request`** (`str`, optional) – The amount of memory (RAM) requested for the deployment in KiB, MiB and GiB. The value must be a string with

a number followed by a unit, for example "512MiB" or "1GiB". This is only available for "low_latency" deployments mode.

- **gpu_request** (int, optional) – The number of GPUs requested for the deployment. This is only available for "low_latency" deployments mode.
- **wait_for_completion** (bool, optional) – Whether to wait for the deployment to be ready. Defaults to True.
- **timeout_s** (int) – Maximum time (in seconds) to wait for the deployment to be ready. Set to None to wait indefinitely. Defaults to None. Only applicable if wait_for_completion is True.

Returns

Created deployment represented as a dict with the following keys:

- "name" (str): Name of the deployment.
- "mode" (str): The deployment mode. Can be "elastic" or "low_latency".
- "pipeline" (dict): Pipeline associated to the deployment represented as dict with the following keys:
  - "name" (str): Name of the pipeline.
- "inputs_mapping" (list of dict): List of inputs mapping represented as dict with the following keys:
  - "pipeline_input_name" (str): Name of the step input.
  - "data_type" (str): Data type of the step input.
  - "description" (str): Description of the step input.
  - "constant_value" (str): Constant value of the step input. Note that this key is only returned if the step input is mapped to a constant value.
  - "environment_variable_name" (str): Name of the environment variable. Note that this key is only returned if the step input is mapped to an environment variable.
  - "endpoint_input_name" (str): Name of the endpoint input. Note that this key is only returned if the step input is mapped to an endpoint input.
  - "is_null" (bool): Whether the step input is mapped to null. Note that this key is only returned if the step input is mapped to null.
  - "datastore_path" (str): Datastore path of the step input. Note that this key is only returned if the step input is mapped to the datastore.
  - "is_required" (bool): Whether the step input is required. Note that this key is only returned if the step input is required.
  - "default_value" (str): Default value of the step input. Note that this key is only returned if the step input has a default value.
- "outputs_mapping" (list of dict): List of outputs mapping represented as dict with the following keys:
  - "pipeline_output_name" (str): Name of the step output.
  - "data_type" (str): Data type of the step output.
  - "description" (str): Description of the step output.

  - `"endpoint_output_name"` (`str`): Name of the endpoint output. Note that this key is only returned if the step output is mapped to an endpoint output.
  - `"is_null"` (`bool`): Whether the step output is mapped to null. Note that this key is only returned if the step output is mapped to null.
  - `"datastore_path"` (`str`): Datastore path of the step output. Note that this key is only returned if the step output is mapped to the datastore.
- `"endpoint_token"` (`str`): Token of the endpoint. Note that this key is only returned if the deployment is an endpoint.
- `"schedule"` (`str`): Schedule of the deployment. Note that this key is only returned if the `execution_rule``of the deployment is ``"periodic"`.
- `"human_readable_schedule"` (`str`): Human readable schedule of the deployment. Note that this key is only returned if the `execution_rule` of the deployment is `"periodic"`.
- `"created_at"` (`str`): Date of creation of the deployment.
- `"created_by"` (`str`): ID of the user who created the deployment.
- `"updated_at"` (`str`): Date of last update of the deployment.
- `"updated_by"` (`str`): ID of the user who last updated the deployment.
- `"last_execution_id"` (`str`): ID of the last execution of the deployment.
- `"is_enabled"` (`bool`): Whether the deployment is enabled.
- `"description"` (`str`): Description of the deployment.
- `"execution_rule"` (`str`): Execution rule of the deployment.
- `"status"` (`str`): The deployment status. Can be `"creation_pending"`, `"up"`, `"creation_failed"`, `"down_retrying"` or `"standby"`.
- `"enable_parallel_executions"` (`bool`): Indicates whether multiple executions can run concurrently within the same pod. This key is only returned if the deployment mode is `"low_latency"`. For more detailed information, see *enable_parallel_executions* parameter of *`create_deployment()`* method.
- `"max_parallel_executions_per_pod"` (`int`): Maximum number of executions that can run at the same time on a deployment's pod in `"low_latency"` mode. This key is only returned if the deployment mode is `"low_latency"` and if `"enable_parallel_executions"` is `True`. For more detailed information, see *max_parallel_executions_per_pod* parameter of *`create_deployment()`* method.
- `"pods"` (`list` of `dict`): List of pods associated with the low latency deployment. Note that this key is only returned if the deployment is in low latency mode. Each pod is represented as `dict` with the following keys:
  - `pod_id` (`str`): ID of the pod.
  - `status` (`str`): Status of the pod.

Return type
`dict[str, str]`

Note: When `execution_rule` is `"endpoint"`:

- When the endpoint deployment is created, it creates an HTTP endpoint which can be called at `POST {environment_url}/endpoints/{endpoint_url_path}`. You can get *environment_url* with *sdk.base_environment_url*. *endpoint_url_path* is the *endpoint_name* if *enpoint_url_path* is not provided.
- Only one step input can be mapped to an endpoint as an `"endpoint_input_name"` if it is of type `"file"`.
- Only one step output can be mapped to an endpoint as an `"endpoint_output_name"` if it is of type `"file"`.

An endpoint token is required to call the HTTP endpoint. This token is different from the SDK token, you can find it in the result of this function as "endpoint_token". Use it to set the `"Authorization"` header as `"EndpointToken {endpoint_token}"`.
Input can be passed to the endpoint either:

- When there is no input file mapped to the endpoint, in the body in JSON with the *application/json* content type and with the format

```
{
  "{input_name}": "{input_value}"
}
```

- When a file input is mapped to the endpoint, as a file with a *multipart/form-data* content type.

This will return the output as:

- When there is no output file, in the body in JSON in the format

```
{
  "output": {
    "{output_name}": "{output_value}"
  }
}
```

- When a file output is mapped, it will return the file as a response with the *application/octet-stream* content type.

A successful call will return results with a status code 200 after redirections.
If your HTTP client does not follow redirection automatically, redirections are indicated by a status code between 300 and 399, and the redirection URL is in the *Location* header. Keep calling the redirection URL until you get a non-redirection status code.
If an endpoint deployment's execution encounters an error, it will return a status code between 400 and 599, and an error message in the body at the property *message*.
Here is an example of a successful call with curl:

```
curl -L "{environment_url}/endpoints/{endpoint_name}" \
  -H "Authorization: EndpointToken {endpoint_token}" \
  -H "Content-Type: application/json; charset=utf-8" \
  -d @- << EOF
{"input_string": "value_1", "input_number": 0}
EOF
```

(continues on next page)

(continued from previous page)

```
# The response will be:
# {"output": {"output_string": "returned_value_1", "output_number": 1}}
```

CraftAiSdk.**list_deployments**()

Get the list of all deployments.

Returns

List of deployments represented as `dict` with the following keys:

- `"name"` (`str`): Name of the deployment.
- `"pipeline_name"` (`str`): Name of the pipeline associated to the deployment.
- `"execution_rule"` (`str`): Execution rule of the deployment. Can be `"endpoint"`, `"run"` or `"periodic"`.
- `"mode"` (`str`): Mode of the deployment. Can be `"elastic"` or `"low_latency"`.
- `"is_enabled"` (`bool`): Whether the deployment is enabled.
- `"created_at"` (`str`): Date of creation of the deployment.

Return type

`list` of `dict`

CraftAiSdk.**get_deployment**(*deployment_name*, *wait_for_completion=False*, *timeout_s=None*)

Get information of a deployment.

Parameters

- **deployment_name** (`str`) – Name of the deployment.
- **wait_for_completion** (`bool`, optional) – Whether to wait for the deployment to be ready. Defaults to `False`.
- **timeout_s** (`int`, optional) – Maximum time (in seconds) to wait for the deployment to be ready. Set to None to wait indefinitely. Defaults to None. Only applicable if `wait_for_completion` is `True`.

Returns

Deployment information represented as `dict` with the following keys:

- `"name"` (`str`): Name of the deployment.
- `"mode"` (`str`): The deployment mode. Can be `"elastic"` or `"low_latency"`.
- `"pipeline"` (`dict`): Pipeline associated to the deployment represented as `dict` with the following keys:
  - `"name"` (`str`): Name of the pipeline.
- `"inputs_mapping"` (`list` of `dict`): List of inputs mapping represented as `dict` with the following keys:
  - `"pipeline_input_name"` (`str`): Name of the step input.
  - `"data_type"` (`str`): Data type of the step input.
  - `"description"` (`str`): Description of the step input.

  - `"constant_value"` (`str`): Constant value of the step input. Note that this key is only returned if the step input is mapped to a constant value.
  - `"environment_variable_name"` (`str`): Name of the environment variable. Note that this key is only returned if the step input is mapped to an environment variable.
  - `"endpoint_input_name"` (`str`): Name of the endpoint input. Note that this key is only returned if the step input is mapped to an endpoint input.
  - `"is_null"` (`bool`): Whether the step input is mapped to null. Note that this key is only returned if the step input is mapped to null.
  - `"datastore_path"` (`str`): Datastore path of the step input. Note that this key is only returned if the step input is mapped to the datastore.
  - `"is_required"` (`bool`): Whether the step input is required. Note that this key is only returned if the step input is required.
  - `"default_value"` (`str`): Default value of the step input. Note that this key is only returned if the step input has a default value.
- `"outputs_mapping"` (`list` of `dict`): List of outputs mapping represented as `dict` with the following keys:
  - `"pipeline_output_name"` (`str`): Name of the step output.
  - `"data_type"` (`str`): Data type of the step output.
  - `"description"` (`str`): Description of the step output.
  - `"endpoint_output_name"` (`str`): Name of the endpoint output. Note that this key is only returned if the step output is mapped to an endpoint output.
  - `"is_null"` (`bool`): Whether the step output is mapped to null. Note that this key is only returned if the step output is mapped to null.
  - `"datastore_path"` (`str`): Datastore path of the step output. Note that this key is only returned if the step output is mapped to the datastore.
- `"endpoint_token"` (`str`): Token of the deployment. Note that this key is only returned if the `execution_rule` of the deployment is `"endpoint"`.
- `"endpoint_url_path"` (`str`): URL path of the deployment. Note that this key is only returned if the `execution_rule` of the deployment is `"endpoint"`.
- `"schedule"` (`str`): Schedule of the deployment. Note that this key is only returned if the `execution_rule` of the deployment is `"periodic"`.
- `"human_readable_schedule"` (`str`): Human readable schedule of the deployment. Note that this key is only returned if the `execution_rule` of the deployment is `"periodic"`.
- `"created_at"` (`str`): Date of creation of the deployment.
- `"created_by"` (`str`): ID of the user who created the deployment.
- `"updated_at"` (`str`): Date of last update of the deployment.
- `"updated_by"` (`str`): ID of the user who last updated the deployment.
- `"last_execution_id"` (`str`): ID of the last execution of the deployment.
- `"is_enabled"` (`bool`): Whether the deployment is enabled.
- `"description"` (`str`): Description of the deployment.

- `"execution_rule"` (`str`): Execution rule of the deployment.
- `"status"` (`str`): The deployment status. Can be `"creation_pending"`, `"up"`, `"creation_failed"`, `"down_retrying"` or `"standby"`.
- `"enable_parallel_executions"` (`bool`): Indicates whether multiple executions can run concurrently within the same pod. This key is only returned if the deployment mode is `"low_latency"`. For more detailed information, see *enable_parallel_executions* parameter of `create_deployment()` method.
- `"max_parallel_executions_per_pod"` (`int`): Maximum number of executions that can run at the same time on a deployment's pod in `"low_latency"` mode. This key is only returned if the deployment mode is `"low_latency"` and if `"enable_parallel_executions"` is `True`. For more detailed information, see *max_parallel_executions_per_pod* parameter of `create_deployment()` method.
- `"pods"` (`list` of `dict`): List of pods associated with the low latency deployment. Note that this key is only returned if the deployment is in low latency mode. Each pod is represented as `dict` with the following keys:
  - `pod_id` (`str`): ID of the pod.
  - `status` (`str`): Status of the pod.

Return type
`dict`

`CraftAiSdk.`**`get_deployment_logs`**(*deployment_name*, *from_datetime=None*, *to_datetime=None*, *limit=None*)

Get the logs of a deployment with `"low_latency"` mode.

Parameters

- **`deployment_name`** (`str`) – Name of the deployment.
- **`from_datetime`** (`datetime.time`, optional) – Datetime from which the logs are collected. If not specified, logs are collected from the beginning.
- **`to_datetime`** (`datetime.time`, optional) – Datetime until which the logs are collected. If not specified, logs are collected until the end.
- **`limit`** (`int`, optional) – Maximum number of logs that are collected. If not specified, all logs are collected.

Returns

List of logs represented as `dict` with the following keys:

- `"timestamp"` (`str`): Timestamp of the log.
- `"message"` (`str`): Message of the log.
- `"stream"` (`str`): Stream of the log. Typically, `"stdout"` or `"stderr"`
- `"pod_id"` (`str`): ID of the pod.
- `"type"` (`str`): Type of the log. Can be `"deployment"`.

Return type
`list` of `dict`

`CraftAiSdk.`**`update_deployment`**(*deployment_name*, *is_enabled=None*, *inputs_mapping=None*, *outputs_mapping=None*, *schedule=None*, *wait_for_completion=True*, *timeout_s=None*)

Update the specified properties of a deployment. The properties that can be updated include enabling/disabling the deployment, updating input/output values, and changing the deployment schedule. Only one property can be updated at a time.

Parameters

- **deployment_name** (str) – Name of the deployment to update.
- **is_enabled** (bool, optional) – Whether the deployment should be enabled or disabled. Disabling a deployment prevents new executions from being triggered. It also frees up computing resources associated to a low-latency deployment. Defaults to *None*.
- **inputs_mapping** (list of instances of *InputSource*) – List of inputs mapping to update with keys *pipeline_input_name*, and *constant_value* or *datastore_path*. The mapping types can not be changed; only the values of constant values (*constant_value*) and data store paths (*datastore_path*) can be updated. See *InputSource* for more details.
- **outputs_mapping** (list of instances of *OutputDestination*) – List of outputs mapping to update with keys *pipeline_output_name* and *datastore_path*. The mapping types can not be changed; only the values of data store paths (*datastore_path*) can be updated. See *OutputDestination* for more details.
- **schedule** (str, optional) – New schedule to be assigned to the periodic deployment. Must be a valid CRON expression. Defaults to *None*.

Returns

Deployment information represented as dict as described in *get_deployment()*.

Return type

dict

CraftAiSdk.**delete_deployment**(*deployment_name*)

Delete a deployment identified by its name.

Parameters

deployment_name (str) – Name of the deployment.

Returns

Deleted deployment represented as dict with the following keys:

- "name" (str): Name of the deployment.
- "execution_rule" (str): Execution rule of the deployment. Can be "endpoint", "run" or "periodic".

Return type

dict

class craft_ai_sdk.**InputSource**(*pipeline_input_name=None*, *endpoint_input_name=None*, *environment_variable_name=None*, *is_required=None*, *default_value=None*, *constant_value=None*, *is_null=None*, *datastore_path=None*, *step_input_name=None*)

Class to specify to which source a step input should be mapped when creating a deployment (cf. *CraftAiSdk.create_deployment()*). The different sources can be one of:

- endpoint_input_name (str): An endpoint input with the provided name.
- constant_value: A constant value.

- `environment_variable_name`: The value of the provided environment variable.
- `is_null`: Nothing.

If the execution rule of the deployment is endpoint and the input is directly mapped to an endpoint input, two more parameters can be specified:

- `default_value`
- `is_required`

Parameters

- **pipeline_input_name** (`str`) – Name of the pipeline input to be mapped.
- **endpoint_input_name** (`str`, optional) – Name of the endpoint input to which the input is mapped.
- **environment_variable_name** (`str`, optional) – Name of the environment variable to which the input is mapped.
- **constant_value** (`Any`, optional) – A constant value.
- **is_null** (`True`, optional) – If specified, the input is not provided any value at execution time.
- **datastore_path** (`str`, optional) – Path of the input file in the datastore. If you want to use a file from the datastore as input, this file will then be accessible as if you passed the file path as an argument to the step. The resulting input will be a `dict` with *"path"* as key and the file path as value. The file will be downloaded in the execution environment before the step is executed. You can then use the file as you would use any other file in the execution environment. Here is an example of how to use this feature in the step code:

```
def step_function(input):
    with open(input["path"]) as f:
        content = f.read()
        print(content)
```

- **default_value** (`Any`, optional) – This parameter could only be specified if the parameter *endpoint_input_name* is specified.
- **is_required** (`bool`, optional) – This parameter could only be specified if the parameter *endpoint_input_name* is specified. If set to *True*, the corresponding endpoint input should be provided at execution time.

class craft_ai_sdk.**OutputDestination**(*pipeline_output_name=None*, *endpoint_output_name=None*, *is_null=None*, *datastore_path=None*, *step_output_name=None*)

Class to specify to which destination a step output should be mapped when creating a deployment (cf. *CraftAiSdk.create_deployment()*). If the execution rule of the deployment is endpoint, an output could either be exposed as an output of the endpoint (via *endpoint_output_name* parameter) or not (via *is_null* parameter).

Parameters

- **pipeline_output_name** (`str`) – Name of the pipeline output to be mapped.
- **endpoint_output_name** (`str`, optional) – Name of the endpoint output to which the output is mapped.
- **is_null** (`True`, optional) – If specified, the output is not exposed as a deployment output.

- **datastore_path** (str, optional) – Path of the output file in the datastore. If you want to upload a file to the datastore as output, you can specify this parameter. The file will be uploaded to the datastore after the step is executed. In order to pass the file to be uploaded in the datastore, you will have to do the same as if you were passing a file as output. You will have to return a dict with *"path"* as key and the file path as value. The file will be uploaded to the datastore after the step is executed. Here is an example of how to use this feature in the step code:

```
def step_function():
    file_path = "path/to/file"
    with open(file_path, "w") as f:
        f.write("content")
    return {"output_file": {"path": file_path}}
```

  You can also specify a dynamic path for the file to be uploaded by using one of the following patterns in your datastore path:

  - *{execution_id}*: The execution id of the deployment.
  - *{date}*: The date of the execution in truncated ISO 8601 (*YYYYMMDD*) format.
  - *{date_time}*: The date of the execution in ISO 8601 (*YYYYMMDD_hhmmss*) format.

class craft_ai_sdk.**DEPLOYMENT_EXECUTION_RULES**(*value*, *names=<not given>*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Enumeration for deployments execution rules.

class craft_ai_sdk.**DEPLOYMENT_MODES**(*value*, *names=<not given>*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Enumeration for deployments modes.

2.5 Endpoint

CraftAiSdk.**trigger_endpoint**(*endpoint_name*, *endpoint_token*, *inputs={}*, *wait_for_completion=True*)

Trigger an endpoint.

Parameters

- **endpoint_name** (str) – Name of the endpoint.
- **endpoint_token** (str) – Token to access endpoint.
- **inputs** (dict, optional) – Dictionary of inputs to pass to the endpoint with input names as keys and corresponding values as values. For files, the value should be an instance of io.IOBase. For json, string, number, boolean and array inputs, the size of all values should be less than 0.06MB. Defaults to {}.
- **wait_for_completion** (bool, optional) – Automatically call *retrieve_endpoint_results* and returns the execution result. Defaults to *True*.

Returns

Created pipeline execution represented as dict with the following keys:

- `"execution_id"` (`str`): ID of the execution.
- `"outputs"` (`dict`): Dictionary of outputs of the pipeline with output names as keys and corresponding values as values. Note that this key is only returned if `wait_for_completion` is *True*.

Return type
`dict`

`CraftAiSdk.`**`retrieve_endpoint_results`**(*endpoint_name*, *execution_id*, *endpoint_token*)

Get the results of an endpoint execution.

Parameters

- **`endpoint_name`** (`str`) – Name of the endpoint.
- **`execution_id`** (`str`) – ID of the execution returned by *trigger_endpoint*.
- **`endpoint_token`** (`str`) – Token to access endpoint.

Returns

Created pipeline execution represented as `dict` with the following keys:

- `"outputs"` (`dict`): Dictionary of outputs of the pipeline with output names as keys and corresponding values as values.

Return type
`dict`

`CraftAiSdk.`**`generate_new_endpoint_token`**(*endpoint_name*)

Generate a new endpoint token for an endpoint.

Parameters
`endpoint_name` (`str`) – Name of the endpoint.

Returns

New endpoint token represented as `dict` with the following keys:

- `"endpoint_token"` (`str`): New endpoint token.

Return type
`dict[str, str]`

2.6 Data store

`CraftAiSdk.`**`upload_data_store_object`**(*filepath_or_buffer*, *object_path_in_datastore*)

Upload a file as an object into the data store.

Parameters

- **`filepath_or_buffer`** (`str`, or file-like object) – String, path to the file to be uploaded ; or file-like object implementing a `read()` method (e.g. via builtin `open` function). The file object must be opened in binary mode, not text mode.
- **`object_path_in_datastore`** (`str`) – Destination of the uploaded file.

`CraftAiSdk.`**`download_data_store_object`**(*object_path_in_datastore*, *filepath_or_buffer*)

Download an object in the data store and save it into a file.

Parameters

- **`object_path_in_datastore`** (`str`) – Location of the object to download from the data store.

- **filepath_or_buffer** (str or file-like object) – String, filepath to save the file to ; or a file-like object implementing a write() method, (e.g. via builtin open function). The file object must be opened in binary mode, not text mode.

Returns
None

CraftAiSdk.**get_data_store_object_information**(*object_path_in_datastore*)

Get information about a single object in the data store.

Parameters
object_path_in_datastore (str) – Location of the object in the data store.

Returns

Object information, with the following keys:

- "path" (str): Location of the object in the data store.
- "last_modified" (str): The creation date or last modification date in ISO format.
- "size" (str): The size of the object.

Return type
dict

CraftAiSdk.**list_data_store_objects**()

Get the list of the objects stored in the data store.

Returns

List of objects in the data store represented as dict with the following keys:

- "path" (str): Location of the object in the data store.
- "last_modified" (str): The creation date or last modification date in ISO format.
- "size" (int): The size of the object in bytes.

Return type
list of dict

CraftAiSdk.**delete_data_store_object**(*object_path_in_datastore*)

Delete an object on the datastore.

Parameters
object_path_in_datastore (str) – Location of the object to be deleted in the data store.

Returns

Deleted object represented as dict with the following keys:

- path (str): Path of the deleted object.

Return type
dict

2.7 Environment Variables

CraftAiSdk.**create_or_update_environment_variable**(*environment_variable_name*, *environment_variable_value*)

Create or update an environment variable available for all pipelines executions.

Parameters

- **environment_variable_name** (str) – Name of the environment variable to create.
- **environment_variable_value** (str) – Value of the environment variable to create.

Returns

None

CraftAiSdk.**list_environment_variables**()

Get a list of all environments variables.

Returns

List of environment variable represented as dict with the following keys:

- name (str): Name of the environment variable.
- value (str): Value of the environment variable.

Return type

list of dict

CraftAiSdk.**delete_environment_variable**(*environment_variable_name*)

Delete the specified environment variable

Parameters

environment_variable_name (str) – Name of the environment variable to delete.

Returns

Deleted environment variable represented as dict with the following keys:

- name (str): Name of the environment variable.
- value (str): Value of the environment variable.

Return type

dict

2.8 Pipeline Metrics

CraftAiSdk.**record_metric_value**(*name*, *value*)

Create or update a pipeline metric. Note that this function can only be used inside a step code.

Parameters

- **name** (str) – Name of the metric to store.
- **value** (float) – Value of the metric to store.

Returns

True if sent, False otherwise

CraftAiSdk.**record_list_metric_values**(*name*, *values*)

Add values to a pipeline metric list. Note that this function can only be used inside a step code.

Parameters

- **name** (str) – Name of the metric list to add values.
- **values** (list of float or float) – Values of the metric list to add.

Returns

True if sent, False otherwise

CraftAiSdk.**get_metrics**(*name=None*, *pipeline_name=None*, *deployment_name=None*, *execution_id=None*)

Get a list of pipeline metrics. Note that only one of the parameters (pipeline_name, deployment_name, execution_id) can be set.

Parameters

- **name** (str, optional) – Name of the metric to retrieve.
- **pipeline_name** (str, optional) – Filter metrics by pipeline, defaults to all the pipelines.
- **deployment_name** (str, optional) – Filter metrics by deployment, defaults to all the deployments.
- **execution_id** (str, optional) – Filter metrics by execution, defaults to all the executions.

Returns

List of execution metrics as dict with the following keys:

- name (str): Name of the metric.
- value (float): Value of the metric.
- created_at (str): Date of the metric creation.
- execution_id (str): Name of the execution the metric belongs to.
- deployment_name (str): Name of the deployment the execution belongs to.
- pipeline_name (str): Name of the pipeline the execution belongs to.

Return type

list of dict

CraftAiSdk.**get_list_metrics**(*name=None*, *pipeline_name=None*, *deployment_name=None*, *execution_id=None*)

Get a list of pipeline metric lists. Note that only one of the parameters (pipeline_name, deployment_name, execution_id) can be set.

Parameters

- **name** (str, optional) – Name of the metric list to retrieve.
- **pipeline_name** (str, optional) – Filter metric lists by pipeline, defaults to all the pipelines.
- **deployment_name** (str, optional) – Filter metric lists by deployment, defaults to all the deployments.
- **execution_id** (str, optional) – Filter metric lists by execution, defaults to all the executions.

Returns

List of execution metric lists as `dict` with the following keys:

- `name` (`str`): Name of the metric.
- `value` (`float`): Value of the metric.
- `created_at` (`str`): Date of the metric creation.
- `execution_id` (`str`): Name of the execution the metric belongs to.
- `deployment_name` (`str`): Name of the deployment the execution belongs to.
- `pipeline_name` (`str`): Name of the pipeline the execution belongs to.

Return type

`list` of `dict`

2.9 Resource Metrics

`CraftAiSdk.`**`get_resource_metrics`**(*start_date*, *end_date*, *csv=False*)

Get resource metrics of the environment.

Parameters

- **`start_date`** (`datetime.datetime`) – The beginning of the period.
- **`end_date`** (`datetime.datetime`) – The end of the period.
- **`csv`** (`bool`) – If True, it will return a csv file as bytes.

Returns

If csv is True, it will return `bytes`. Otherwise: `dict`: The resource metrics, with the following keys:

- `additional_data` (`dict`): Additional data with the following keys:
  - `total_disk` (`int`): Total disk size in bytes.
  - `total_ram` (`int`): Total RAM size in bytes.
  - `total_vram` (`int`): Total VRAM size in bytes if there is a GPU.
- `metrics` (`dict`): The metrics of the environment with the following keys:
  - `cpu_usage` (`list` of `dict`): The CPU usage in percent.
  - `disk_usage` (`list` of `dict`): The disk usage in percent.
  - `ram_usage` (`list` of `dict`): The RAM usage in percent.
  - `vram_usage` (`list` of `dict`): The VRAM usage in percent if there is a GPU.
  - `gpu_usage` (`list` of `dict`): The GPU usage in percent if there is a GPU.
  - `network_input_usage` (`list` of `dict`): The network input usage in bytes.
  - `network_output_usage` (`list` of `dict`): The network output usage in bytes.

Each element of the lists is a dict with the following keys:

- `metric` (`dict`): Dictionary with the following key:
  - `worker` (`str`): The worker name.

- `values` (`list` of `list`): The values of the metrics in the following format: `[[timestamp, value], ...]`.

2.10 Users

`CraftAiSdk.`**`get_user`**`(`*`user_id`*`)`

Get information about a user.

Parameters

`user_id` (`str`) – The id of the user.

Returns

The user information, with the following keys:

- `id` (`str`): id of the user.
- `name` (`str`): Name of the user.
- `email` (`str`): Email of the user.

Return type

`dict`

2.11 Vector Database

`CraftAiSdk.`**`get_vector_database_credentials`**`()`

Get the credentials of the vector database.

Returns

The vector database credentials, with the following keys:

- `"vector_database_url"` (`str`): URL of the vector database.
- `"vector_database_token"` (`str`): Token to connect to the vector database.

Return type

`dict`

Python Module Index

C

Index

Symbols

A

B

C

D

F

G