Cisco Edge Intelligence local manager CLI Utility

Cisco Edge Intelligence CLI utility tool

This document takes you through an example of a CLI utility tool designed for Cisco Edge Intelligence, Release 2.0. This utility tool example is available on GitHub, and offers guidance on how you can perform bulk pipeline operations. See the Readme file for instructions on how to build, install, and use the CLI package in your local system.

The tool contains JSON files that provide examples of pipeline templates, and data logic, data source, data target, and data variable configurations.

You can deploy one or more pipelines on one or more devices, by creating the necessary inventory files in the YAML format.

The CLI utility tool supports the following functions:

- Change device passwords.
- Create pipeline inventories.
- Deploy or undeploy pipelines.
- Check the health status of pipelines.
- View event logs by operation type and timestamp for troubleshooting.

Download and install the Cisco Edge Intelligence CLI package

Before you begin

- Python release 3.10 or later is required on your system.
- To use the CLI tool, make sure that python bin folder is available to the system path.
- Active Internet access is essential to download the Cisco Edge Intelligence CLI package.

Procedure

Step 1 Download the contents of the CLI utility tool for Cisco Edge Intelligence to your local system.

Step 2 To build the CLI utility tool package, use the following commands:

```
cd ei-utility-tools
python3 setup.py sdist bdist_wheel
```

Step 3 To install the package, use the following commands:

```
cd ei-utility-tools/dist
pip install eilm-cli-1.1.0.tar.gz
```

Configure and Initialize Cisco Edge Intelligence local manager CLI

Carry out the following procedure to set a home directory environment. We will initiate the command line and generate the configuration files in the environment.

Procedure

Step 1 To create an environment, create a directory **eilm-cli** in the Home folder.

Step 2 Set the path for environment variable (EI_CLI_HOME=<path_to_eilm_home_dir>),

- For Mac or Linux command line tools, use the **export** command followed by the destination path.

  ```
  export EI_CLI_HOME=/home/user/eilm-cli
  ```

- For Windows Powershell:

  ```
  $env:EI_CLI_HOME="C:\Users\Administrator\eilm-cli"
  ```

  Note
  In Windows Powershell, the path must be enclosed in double quotations.

For Windows CMD

```
set EI_CLI_HOME="C:\Users\Administrator\eilm-cli"
```

or

```
set EI_CLI_HOME=C:\Users\Administrator\eilm-cli
```

Step 3 To generate a configuration folder and sample configuration files, enter **eilm-cli init**.

```
~/eilm-cli$ eilm-cli init
EI CLI env successfully initialized at '/Users/EI/eilm-cli'
```

Step 4 A folder with the name **configs** is created in the eilm-cli folder. The configs folder contains example JSON files for data logic, data source, and data target configurations.

Example:

```
~/eilm-cli$ ls -ltr
total 2
drwxr-xr-x  3 root  root    96 Jan 11 15:45 configs
-rw-r--r--  1 root  root  1029 Jan 11 15:55 eilm-cli.log
~/EI/eilm-cli$ ls -ltr configs/sample
total 4
-rw-r--r--  1 root  root   493 Jan 11 15:43 data_logic_1.json
-rw-r--r--  1 root  root  1359 Jan 11 15:43 data_source_1.json
-rw-r--r--  1 root  root   968 Jan 11 15:43 data_target_1.json
-rw-r--r--  1 root  root   119 Jan 11 15:43 data_vars_1.json
```

Step 5 To verify the functionality of eilm-cli utility tool, use the **help** command.

```
~/eilm-cli$ eilm-cli --help
usage: eilm-cli [-h] {init,user,agent,pipeline} ...
EI Local Manager CLI Orchestrator [Version 1.1.0]
positional arguments:
  {init,user,agent,pipeline}
                        options
    init                initialize eilm-cli environment
    user                user operations (change-password)
    agent               agent operations (status, reset)
    pipeline            pipeline operations (deploy, undeploy, status)
options:
  -h, --help            show this help message and exit
```

Create a pipeline inventory file

To create a pipeline configuration file (YAML file), you must first create individual configuration files (JSON files) defining data source, data logic, data target, and data variables. Then, create the inventory file that brings the various configurations together as a pipeline that can be deployed.

You can reuse the individual JSON configuration files in multiple pipelines, as needed.

You can create as many inventory files as needed.

The following taskflow contains examples of the configuration files that are available through the eilm-cli utility tool. To obtain the exact configuration for a pipeline, export the pipeline template from your Cisco Edge Intelligence GUI.

Procedure

Step 1 Use the exported pipeline template as a template to deploy on other EI agents. The file contains the data source, destination, and pipeline configuration.

```
{
  "name": "pipeline-template-datalogic",
  "data": {
    "dataSources": {
      "mqtt1": {
        "type": "MQTT",
        "assetId": "1234",
        "dataSourceConfiguration": {
          "port": null,
          "secure": false,
```

```
            "useWebSockets": false,
            "clientId": "test",
            "assetId": "1234",
            "name": "endpoint_1883",
            "endpoints": [
              {
                "clientId": "test",
                "port": "1883",
                "credentials": [
                  {
                    "username": "gogopals@cisco.com",
                    "password":
"10000:KusMo5cIWx/tkKNKeeDfWg==:+z5Pjl3JkNgqLje+Hl+xDnhG7jEL04/F0LyAyIY1KO3avuZn8n8k/+2xchX9s6ty/W4azkJbb1nnflUaeh4YCg=="

                  }
                ],
                "name": "endpoint_1883",
                "metrics": {
                  "test": {
                    "name": "test",
                    "label": "test",
                    "datatype": "String",
                    "topic": "test_topic"
                  },
                  "test2": {
                    "name": "test2",
                    "label": "test2",
                    "datatype": "String",
                    "topic": "test_topic2"
                  }
                },
                "secure": false,
                "privateKey": null,
                "serverCertificate": null
              }
            ]
          },
          "fields": [
            {
              "type": "subscription",
              "key": "test",
              "source": "/endpoints/endpoint_1883/test/test_topic"
            },
            {
              "type": "subscription",
              "key": "test2",
              "source": "/endpoints/endpoint_1883/test/test_topic2"
            },
            {
              "type": "customAttribute",
              "key": "custom1",
              "value": "test_custome_value"
            }
          ]
        },
        "mqtt2": {
          "type": "MQTT",
          "assetId": "4567",
          "dataSourceConfiguration": {
            "port": null,
            "secure": false,
            "useWebSockets": false,
            "clientId": "gg-mclient-2",
            "assetId": "4567",
```

```
          "name": "endpoint_1883",
          "endpoints": [
            {
              "clientId": "gg-mclient-2",
              "port": "1883",
              "credentials": [
                {
                  "username": "gogopals@cisco.com",
                  "password":
"10000:AZhjbaVaNp7imy7ZYvVRRQ==:d/pMLrrCfUKuJbTbKUgplumM0THVvJjZD5s6Mw0nhx4z+vptlTp4glkV5j57l7p1tIJFDpOq/V9q+nU5jPXKzQ=="

                }
              ],
              "name": "endpoint_1883",
              "metrics": {
                "test2": {
                  "name": "test2",
                  "label": "test",
                  "datatype": "String",
                  "topic": "test_topic2"
                }
              },
              "secure": false,
              "privateKey": null,
              "serverCertificate": null
            }
          ]
        },
        "fields": [
          {
            "type": "subscription",
            "key": "test2",
            "source": "/endpoints/endpoint_1883/gg-mclient-2/test_topic2"
          }
        ]
      }
    },
    "dataTarget": {
      "type": "MQTT",
      "dataTargetConfiguration": {
        "connectors": [
          {
            "host": "$TARGET_HOST",
            "port": 1884,
            "secure": false,
            "verifyPeer": false,
            "mqttQoS": 1,
            "clientId": "12344567_3fcbaece-ce1b-428b-995e-2b0319710180",
            "cleanSession": true,
            "username": "gogopals@cisco.com",
            "password": "Welcome@12345",
            "publish": {
              "resultPath": {
                "mqttRetain": true,
                "mqttQos": 1,
                "mqttTopic": "cisco/edge-intelligence/telemetry/12344567"
              }
            }
          }
        ]
      }
    },
    "outputModel": {},
    "scriptedDataLogic": {
```

```
      "enableCloudCommands": false,
      "parameters": {
        "INVOKE_ON_NEW_DATA": null
      },
      "productive": true,
      "script":
"[illegible]",

      "invokeEvery": 1000
    }
  }
}
```

Step 2 The data_source configuration file defines the sources from which a pipeline collects edge data. You can add up to 20 data sources to a pipeline.

```
{
  "dataSources": {
    "mqtt1": {
      "type": "MQTT",
      "assetId": "1234",
      "dataSourceConfiguration": {
        "port": null,
        "secure": false,
        "useWebSockets": false,
        "clientId": "test",
        "assetId": "1234",
        "name": "endpoint_1883",
        "endpoints": [
          {
            "clientId": "test",
            "port": "1883",
            "credentials": [
              {
                "username": "test_user@cisco.com",
                "password": "Welcome@12345"
              }
            ],
            "name": "endpoint_1883",
            "metrics": {
              "test": {
                "name": "test",
                "label": "test",
                "datatype": "String",
                "topic": "test_topic"
              },
              "test2": {
                "name": "test2",
                "label": "test2",
                "datatype": "String",
                "topic": "test_topic2"
              }
            },
            "secure": false,
            "privateKey": null,
            "serverCertificate": null
          }
        ]
      },
      "fields": [
        {
          "type": "subscription",
          "key": "test",
          "source": "/endpoints/endpoint_1883/test/test_topic"
```

```
        },
        {
          "type": "subscription",
          "key": "test2",
          "source": "/endpoints/endpoint_1883/test/test_topic2"
        },
        {
          "type": "customAttribute",
          "key": "custom1",
          "value": "test_custome_value"
        }
      ]
    },
    "mqtt2": {
      "type": "MQTT",
      "assetId": "4567",
      "dataSourceConfiguration": {
        "port": null,
        "secure": false,
        "useWebSockets": false,
        "clientId": "gg-mclient-2",
        "assetId": "4567",
        "name": "endpoint_1883",
        "endpoints": [
          {
            "clientId": "gg-mclient-2",
            "port": "1883",
            "credentials": [
              {
                "username": "test_user@cisco.com",
                "password": "Welcome@12345"
              }
            ],
            "name": "endpoint_1883",
            "metrics": {
              "test2": {
                "name": "test2",
                "label": "test",
                "datatype": "String",
                "topic": "test_topic2"
              }
            },
            "secure": false,
            "privateKey": null,
            "serverCertificate": null
          }
        ]
      },
      "fields": [
        {
          "type": "subscription",
          "key": "test2",
          "source": "/endpoints/endpoint_1883/gg-mclient-2/test_topic2"
        }
      ]
    }
  }
}
```

Step 3 The data_logic file defines how a data is processed and transformed. The example file includes data rule and data logic configurations for your reference.

```
{
  "scriptedDataLogic": {
```

```
    "enableCloudCommands": false,
    "parameters": {
      "INVOKE_ON_NEW_DATA": null
    },
    "productive": true,
    "script":
"[illegible]",

    "invokeEvery": "$INVOKE_INTERVAL"
  }
}
```

Step 4 The data_target file defines the destinations that the transformed data is delivered. The example file also includes output model configurations.

```
{
  "dataTarget": {
    "type": "MQTT",
    "dataTargetConfiguration": {
      "connectors": [
        {
          "host": "$TARGET_HOST",
          "port": 1884,
          "secure": false,
          "verifyPeer": false,
          "mqttQoS": "$MQTT_QOS",
          "clientId": "12344567_3fcbaece-ce1b-428b-995e-2b0319710180",
          "cleanSession": "$CLEAN_SESSION",
          "username": "test_user@cisco.com",
          "password": "Welcome@12345",
          "publish": {
            "resultPath": {
              "mqttRetain": true,
              "mqttQos": "$MQTT_QOS",
              "mqttTopic": "cisco/edge-intelligence/telemetry/12344567"
            }
          }
        }
      ]
    },
    "downSampling": {
      "update_interval": 1000
    }
  },
  "outputModel": {
    "test": {
      "addTimestamp": false,
      "emit": true,
      "targetPath": "test",
      "type": "STRING"
    },
    "test2": {
      "addTimestamp": false,
      "emit": true,
      "targetPath": "test2",
      "type": "STRING"
    }
  }
}
```

Step 5 The data_vars file defines the template variables used in the pipeline configuration.

```
{
    "INVOKE_INTERVAL": 1000,
```

```
    "MQTT_QOS": 1,
    "CLEAN_SESSION": true,
    "TARGET_HOST": "test.mosquitto.org"
}
```

Step 6 In the configs folder of the elim-cli tool, create an inventory file to define the pipelines you want to deploy. One inventory file can contain pipeline configurations for multiple Cisco Edge Intelligence agents. You can also choose to create multiple inventory files in the configs folder.

```
- DEVICE_IP: 10.1.1.100:8008
  PIPELINE_NAME: pipeline1
  PIPELINE_TEMPLATE: test_pipeline_template.json
  DATA_SOURCE_FILE: data_source_1.json
  DATA_LOGIC_FILE: data_logic_1.json
  DATA_TARGET_FILE: data_target_1.json
  DATA_VARIABLES_FILE: data_vars_1.json
- DEVICE_IP: 10.1.1.101:8008
  PIPELINE_NAME: pipeline2
  PIPELINE_TEMPLATE: test_pipeline_template.json
  DATA_SOURCE_FILE: data_source_2.json
  DATA_LOGIC_FILE: data_logic_2.json
  DATA_TARGET_FILE: data_target_3.json
  DATA_VARIABLES_FILE: data_vars_2.json
- DEVICE_IP: 10.1.1.102:8008
  PIPELINE_NAME: pipeline3
  PIPELINE_TEMPLATE: test_pipeline_template.json
  DATA_SOURCE_FILE: data_source_1.json
  DATA_LOGIC_FILE: data_logic_3.json
  DATA_TARGET_FILE: data_target_2.json
  DATA_VARIABLES_FILE: data_vars_1.json
```

Commands for pipeline operations

The command for pipeline management requires password authentication. For bulk deployment or undeployment of pipelines across multiple devices, the access password across the devices must match.

To change the password across the devices mentioned in your inventory file, use the `eilm-cli user change-pwd -f inventory.yaml` command.

You are prompted to enter the old and new passwords for each device in the inventory file, and the passwords are updated accordingly.

Deploy a Template

To deploy a template, use the `eilm-cli pipeline deploy-template -f test_inventory.yml`command.

Deploy a pipeline

To deploy a pipeline, use the `eilm-cli pipeline deploy -f inventory.yaml` command.

Undeploy a pipeline

To undeploy a pipeline, use the `eilm-cli pipeline undeploy -f inventory.yaml` command.

Check pipeline status

To check the status of a pipeline, use the `eilm-cli pipeline status -f inventory.yaml` command.

Logs Management

The eilm-cli.log folder in the Cisco Edge Intelligence CLI utility tool stores all the log files.

The log files are named based on the operation and timestamp:

- `change_pwd_<timestamp>.log`
- `deploy_<timestamp>.log`
- `status_<timestamp>.log`
- `undeploy_<timestamp>.log`

Troubleshooting

If you encounter issues:

1. Review the log files for error details.
2. Verify the `inventory.yml` file for accuracy.
3. Ensure all dependencies are installed and up to date.

For additional command details, refer to the `eilm-cli --help` command.