Koo Command Line Interface

FAQs

Issue	01
Date	2021-12-17

HUAWEI

HUAWEI TECHNOLOGIES CO., LTD.

Huawei Technologies Co., Ltd.

Address:	Huawei Industrial Base Bantian, Longgang Shenzhen 518129 People's Republic of China
Website:	https://www.huawei.com
Email:	support@huawei.com

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process*. For details about this process, visit the following web page:
https://www.huawei.com/en/psirt/vul-response-process
For vulnerability information, enterprise customers can visit the following web page:
https://securitybulletin.huawei.com/enterprise/en/security-advisory

Contents

1 Overview

Koo Command Line Interface (KooCLI) classifies command errors into five types, each of which has a different identifier at the start of the error message. Troubleshoot the errors as follows:

- [NETWORK_ERROR]: HTTP request exceptions. Check the network connection.
- [CLI_ERROR]: KooCLI exceptions that occur during command processing. Contact the on-call personnel of KooCLI.
- [USE_ERROR]: Errors caused by an incorrect parameter in a command. Modify the parameter according to the error message.
- [OPENAPI_ERROR]: Errors that occur when calling a cloud service API. Contact the on-call personnel of the cloud service.
- [APIE_ERROR]: Errors that occur when calling API Explorer to obtain metadata. Contact the on-call personnel of API Explorer.

Alternatively, you can look through **the following FAQs** to find required information.

Table 1-1 FAQs

Category	Link
Authentication	**How Do I Obtain a Permanent AK/SK?**
	How Do I Obtain an Account ID and Project ID?
	How Do I Obtain a Region?
	How Do I Obtain a Temporary AK/SK and SecurityToken?
	Authentication Mode Priority
Profiles	**Which Profile Will Be Used by Default If No Profile Is Specified in a Command?**
Metadata cache	**Where Are Metadata Cache Files Stored? How Do I Clear Them?**

Category	Link
Logs	Where Are Log Files Stored?
Network connections	What Can I Do If the Network Connection Times Out?
Cloud services	Why Am I Seeing a Message Indicating an Unsupported Service?
Cloud service APIs	Why Am I Seeing a Message Indicating an Unsupported Operation?
	How Do I Specify a Cloud Service API and Its Version?
	How Can I Determine Whether a Command Is Successfully Executed from an Empty Response?
Regions	Why Am I Seeing a Message Indicating that the cli-region Parameter Is Missing?
	Why Am I Seeing a Message Indicating Unsupported cli-region?
Parameters	What Are KooCLI System Parameters?
	Why Am I Seeing a Message Indicating an Invalid Parameter?
	Why Are Old and New System Parameters (Such as region and cli-region) Available? Which Parameters Are Recommended?
	Why Am I Seeing a Message Indicating a Duplicate Parameter?
	How Do I Use cli-jsonInput?
	When Can I Use cli-jsonInput?
	Why Am I Seeing a Message Indicating an Unsupported Parameter Position or Type?
	How Do I Leave a Body Parameter Empty for Cloud Service APIs?
Interactive mode and autocomplete	How Do I Use Interactive Mode and Autocomplete?
Output formats	What Output Formats Are Supported by KooCLI?
	How Do I Use JMESPath Expressions?
	Which Built-in Functions Are Supported by JMESPath?

Category	Link
	Which KooCLI System Parameters Are Related to Data Output? Which Ones Are Recommended?
	How Do I Use cli-output, cli-query, and cli-output-num?
	How Do I Use cli-output-rows, cli-output-cols, and cli-output-num?
	What Are the Precautions for Using cli-output-rows, cli-output-cols, and cli-output-num?
	How Do I Use cli-json-filter?
	What Are the Precautions for Using cli-json-filter?
Other	How Do I Use KooCLI in Non-configuration Mode?
	Should I Enclose a Service Name, Operation, and Parameter Value in Quotation Marks in a Command?
	What Are the Application Scenarios of Online/Offline Modes?
	How Do I Uninstall KooCLI?

2 Authentication

2.1 Obtaining Authentication Information

How Do I Obtain a Permanent AK/SK?

See **Obtaining a Permanent AK/SK**.

How Do I Obtain an Account ID and Project ID?

See **Obtaining an Account ID and Project ID**.

How Do I Obtain a Region?

For details, see **Regions and Endpoints**.

How Do I Obtain a Temporary AK/SK and SecurityToken?

See **Obtaining a Temporary Access Key and SecurityToken Through a Token**.

2.2 Authentication Mode Priority

During command parsing, KooCLI performs authentication for API calling according to the following authentication mode priority:

1. AK/SK authentication in non-configuration mode
   a. Access key (permanent AK/SK, that is, **cli-access-key** and **cli-secret-key**) or temporary security credentials (temporary AK/SK and SecurityToken, that is, **cli-access-key**, **cli-secret-key**, and **cli-security-token**)
2. Profile specified in the command or the default profile
3. Cloud service agency (applicable only when KooCLI is used on an ECS)

If an exception occurs during authentication mode parsing, authentication modes with a lower priority will not be used.

3 Profiles

3.1 Which Profile Will Be Used by Default If No Profile Is Specified in a Command?

Background

When you use KooCLI to manage and use your cloud service resources, it preferentially uses the profile specified by the **--cli-profile** option in the command during an API call.

If no profile is specified in the command, the **default profile** is used to call the target API.

If the profile does not match the target API or a parameter required for access to the target API is missing, an error message is displayed, for example:

- [USE_ERROR] Specify cli-region.
- [USE_ERROR] The value of cli-region is not supported. Supported regions: *
- [USE_ERROR] The following parameters are required: project_id
- [USE_ERROR] cli-domain-id is required for access to global services using AK/SK. Add this parameter or run `hcloud configure set` to configure it.

Solution

- If you specify a profile with **--cli-profile** in the command, run the **hcloud configure show --cli-profile=***${profileName}* command to view the profile, and check whether the specified profile is proper.
- If no profile is specified in the command, the **default profile** is used to call the target API. Run the **hcloud configure show** command to query details about the default profile.
- To use other profiles, run the **hcloud configure list** command to view all the configured profiles, use **--cli-profile=***${profileName}* in the API calling command to specify the name of the target profile, and try again.

4 Metadata Cache

4.1 Where Are Metadata Cache Files Stored? How Do I Clear Them?

When you use KooCLI to manage and use your cloud service resources, it remotely obtains details about the target cloud services and APIs in the commands. To reduce the remote calling times and accelerate response, KooCLI introduces the cache mechanism to store the information about cloud services and their APIs in local cache files, known as metadata cache files. Before the files expire, the included information is used to verify and assemble parameters in commands.

- Storage directory of metadata cache files
  - Online mode
    - Windows: **C:\Users\***{Your Windows username}***\\.hcloud\metaRepo\**
    - Linux: **/home/***{Current username}***/.hcloud/metaRepo/**
    - macOS: **/Users/***{Current username}***/.hcloud/metaRepo/**
  - Offline mode
    - Windows: **C:\Users\***{Your Windows username}***\\.hcloud\metaOrigin\**
    - Linux: **/home/***{Current username}***/.hcloud/metaOrigin/**
    - macOS: **/Users/***{Current username}***/.hcloud/metaOrigin/**
- Procedure for clearing metadata cache files
  - Online mode

    To clear metadata cache files, run **hcloud meta clear**. After they are cleared, data is obtained and written to new cache files during API calling.
  - Offline mode

    Run the **hcloud meta clear** command to clear the metadata cache files parsed from the downloaded offline metadata package. The package will remain. During API calling, this package will be parsed again and new metadata cache files will be written into it.

5 Logs

5.1 Where Are Log Files Stored?

KooCLI records the logs generated for running commands. Currently, log recording cannot be disabled.

Log files are named **hcloud.log** and stored in the following directories:

- Linux: **/home/***{Current username}***/.hcloud/log/**
- Windows: **C:\Users\***{Current username}***\\.hcloud\log\**
- macOS: **/Users/***{Current username}***/.hcloud/log/**

6 Network Connections

6.1 What Can I Do If the Network Connection Times Out?

Background

During a cloud service API call, KooCLI verifies the validity of your input parameters. Specifically, it first remotely obtains the details about the cloud service and API, and then remotely calls the target API. If the request fails due to a network connection error, an error message is displayed. For example:

- [NETWORK_ERROR] Connection timed out. Check network connectivity.
- [USE_ERROR] API calling timed out. Check the value of readTimeout in the profile or command.
- [NETWORK_ERROR] Connection timed out * consecutive times (reconnection attempts: *). Check network connectivity.

Solution

1. Check whether your network connection is normal.
2. If the network connection is normal but the error message indicates that the connection timed out, the value of **cli-connect-timeout** in the profile or command may be too small. Change the value as follows:
   - If the **--cli-connect-timeout** option is used in the command, increase the value and try again.
   - If the **--cli-connect-timeout** option is not used in the command, the value of this parameter in the current profile is used during command execution. Add **--cli-connect-timeout=***${connectTimeout}* to the current command to temporarily overwrite the parameter value in the profile, and try again. To change the parameter value in the profile, run the **hcloud configure set --cli-profile=***${profileName}* **--cli-connect-timeout=***${connectTimeout}* command.

3. If the network connection is normal but the error message prompts you to check **readTimeout**, the value of **cli-read-timeout** in the profile or command may be too small. Change the value as follows:
   – If the **--cli-read-timeout** option is used in the command, increase the value and try again.
   – If the **--cli-read-timeout** option is not used in the command, the value of this parameter in the current profile is used during command execution. Add **--cli-read-timeout=***${readTimeout}* to the current command to temporarily overwrite the parameter value in the profile, and try again. To change the parameter value in the profile, run the **hcloud configure set --cli-profile=***${profileName}* **--cli-read-timeout=***${readTimeout}* command.

7 Cloud Services

7.1 Why Am I Seeing a Message Indicating an Unsupported Service?

Background

During a cloud service API call, KooCLI verifies the validity of your input parameters. If the cloud service name in the command is incorrect, or the cloud service is not released on KooCLI, an error message is displayed:

[USE_ERROR] Unsupported service: *

Solution

1. Run the **hcloud --help** command to view all supported cloud services and check whether the service name is correct.
2. If the service name is correct but the service is not in the **Available services** list output by the **hcloud --help** command, check either of the following reasons:
   a. Using the **online mode**: The cloud service has not been launched on KooCLI.
   b. Using the **offline mode**: The offline data package you used may not be the latest version, resulting in a parameter verification failure. In this case, run the **hcloud meta download** command to update the offline data package, and then run the command in **1** again. If the cloud service still does not exist in the result, it is unavailable for KooCLI offline mode. Try again when the offline data package is updated, or **use the online mode** instead.
3. Determine your current language environment based on the language setting on KooCLI or your system language. Currently, the cloud services released in the English environment are different from those released in the Chinese

environment. To switch the language, run the **hcloud configure set --cli-lang=en** command.

8 Cloud Service APIs

8.1 Why Am I Seeing a Message Indicating an Unsupported Operation?

Background

During a cloud service API call, KooCLI verifies the validity of your input parameters. If the API operation in the command is incorrect or the API is not released on KooCLI, the following error message is displayed:

[USE_ERROR] Unsupported operation: *

Solution

1. Run the **hcloud <service> --help** command to view the list of supported operations and check whether the operation is correct.
2. If the operation is correct but not included in the **Available Operations** list output by the **hcloud <service> --help** command, check either of the following reasons:
   a. Using the **online mode**: The corresponding API has not been launched on KooCLI.
   b. Using the **offline mode**: The offline data package you used may not be the latest version, resulting in a parameter verification failure. In this case, run the **hcloud meta download** command to update the offline data package, and then run the command in **1** again. If the operation still does not exist in the result, the corresponding API is unavailable for KooCLI offline mode. Try again when the offline data package is updated, or **use the online mode** instead.

3. Determine your current language environment based on the language setting on KooCLI or your system language. Currently, the cloud services and APIs released in the English environment are different from those in the Chinese environment. To switch the language, run the **hcloud configure set --cli-lang=en** command.

8.2 How Do I Specify a Cloud Service API and Its Version?

Background

During a cloud service API call, KooCLI verifies the validity of your input parameters. If the service has multiple versions, some or all APIs of the service also have multiple versions. The parameters and application scenarios of an API may vary with versions. The CLI needs to obtain the version information of an API that belongs to a service with multiple versions.

Querying and Specifying the Version of a Cloud Service API

- Querying version information

  Run **hcloud <service> --help** to view the operation list of the cloud service. If an operation appears multiple times in the **Available Operations** list and different version numbers are concatenated with a slash (/), specify a version when calling the API. For details, see **Specifying a version**. For other operations that appear only once in the operation list, their version numbers do not need to be concatenated. KooCLI calls their only version.

- Specifying a version
  - Manually specifying a version

    When calling an API that belongs to a service with multiple versions, use a slash (/) to manually add the version number after **operation** of the original API. For example, if **operation** is **showLogs** in the original command and **showLogs/v1** and **showLogs/v2** exist in the **Available Operations** list, you can set **operation** to **showLogs/v1** or **showLogs/v2**.
  - Using the default version

    When calling an API that belongs to a service with multiple versions, if you do not specify a version in the command, KooCLI obtains all available versions of the API and sorts them in alphabetical order. The last version of the API in the list is called by default. (Generally, this version is the latest version of the API.)

Extended Question and Solution

- Question

  When you call an API that belongs to a service with multiple versions by **using the API's default version**, if the metadata files cached locally are modified, KooCLI may not be able to correctly parse the API version based on the cache files. In this case, the following error message is displayed:

  ```
  [USE_ERROR] The API has multiple versions. Specify one.
  ```

- Solution

  Run **hcloud meta clear** to clear the cache files, and try again.

8.3 How Can I Determine Whether a Command Is Successfully Executed from an Empty Response?

Description

Calling APIs of some cloud services using KooCLI does not print any response. You cannot determine whether the calling is successful.

Solution

Add **--debug** to the original command to print the debugging information during command invocation. This line will be included: "API response status code is xxx." You can check the command invocation status based on the status code.

9 Regions

9.1 Why Am I Seeing a Message Indicating that the cli-region Parameter Is Missing?

Background

When you use KooCLI to call cloud service APIs, a proper value of **cli-region** is required. If you have neither specified the value of **cli-region** in the command, nor configured it in the current profile, the following error message is displayed:

[USE_ERROR] Specify cli-region.

Solution

1. If the value of **cli-region** is not specified in the command, add **--cli-region=***${regionValue}* to the command and try again.
2. If there is a region you often use, run the **hcloud configure set --cli-profile=***${profileName}* **--cli-region=***${regionValue}* command to add it to the target profile. When you use the profile again to call an API, you do not need to enter the value of **cli-region** in the command. However, if the target API is not available in this region, enter **--cli-region=***${regionValue}* in the command to specify a supported region.

9.2 Why Am I Seeing a Message Indicating Unsupported cli-region?

Background

When you use KooCLI to call cloud service APIs, a proper value of **cli-region** is required. If you encounter any of the following situations:

- The value of **cli-region** in the command is incorrect.
- The value of **cli-region** is specified in the command, but the target API does not support the region.
- The value of **cli-region** is not specified in the command, and the target API does not support the region obtained from the current profile.

Either of the following error messages is displayed:

[USE_ERROR] The value of cli-region is not supported. Supported regions: *

[USE_ERROR] Value of cli-region in the current profile is not supported. Supported regions: *

Solution

1. View the supported regions in the prompt and check whether the **cli-region** value you specified is correct.
2. If the value is correct but one of the preceding error messages is displayed during command execution, check the following possible causes:
   a. Using the **online mode**: The target API does not support the **cli-region**. Modify the parameter as follows:
      i. If you have specified the value of **cli-region** in the command, change this value based on the list of supported regions in the prompt message, and try again.
      ii. If the value of **cli-region** is not specified in the command, KooCLI parses and uses the value in the current profile. Add **--cli-region=***${regionValue}* to the current command based on the list of supported regions in the error message, and try again. To change the value of **cli-region** in the profile, run the **hcloud configure set --cli-profile=***${profileName}* **--cli-region=***${regionValue}* command.
   b. Using the **offline mode**: The offline data package you used may not be the latest version, resulting in a parameter verification failure. In this case, run the **hcloud meta download** command to update the offline data package, and run the original command again. If the error persists, the **cli-region** value is unavailable for KooCLI offline mode. Try again when the offline data package is updated, or **use the online mode** instead.
3. If the preceding error messages are not displayed during the command execution, but the returned value of the called API indicates that the region is incorrect, the local cache files containing the **cli-region** list may be modified. As a result, the parameter verification of KooCLI is inaccurate. In this case, run the **hcloud meta clear** command to clear the local cache files, and try again.
4. Determine your current language environment based on the language setting on KooCLI or your system language. Currently, the regions (**cli-region**) supported by cloud service APIs in the English environment are different from those in the Chinese environment. To switch the language, run the **hcloud configure set --cli-lang=en** command.

10 Parameters

10.1 What Are KooCLI System Parameters?

System Parameters

KooCLI system parameters are internal parameters. **The following table** describes the system parameters and their usage.

Table 10-1 KooCLI's new system parameters

Parameter	Description	How to Use
help	Prints help information.	Use it directly in a command.
debug	Prints debugging information.	Use it directly in a command.
dryrun	Prints the request message after verification, without execution.	Use it directly in a command.

Parameter	Description	How to Use
interactive	Puts you into the interactive mode.	Use it directly in a command.
cli-region	Region.	Configure it in a profile or use it directly in a command.
cli-access-key	Access key ID required in AK/SK mode.	Configure it in a profile or use it directly in a command.
cli-secret-key	Secret access key required in AK/SK mode.	Configure it in a profile or use it directly in a command.
cli-domain-id	Account ID.	Configure it in a profile or use it directly in a command.
cli-project-id	Project ID	Configure it in a profile or use it directly in a command.
cli-profile	Profile. If not specified, the default one is used.	Configure it in a profile or use it directly in a command.
cli-mode	Authentication mode (**AKSK** or **ecsAgency**).	Configure it in a profile or use it directly in a command.
cli-jsonInput	Passes API parameters using a JSON file.	Use it directly in a command.
cli-output	Response data output format (**json**, **table**, or **tsv**).	Use it directly in a command.
cli-query	JMESPath for filtering response data.	Use it directly in a command.
cli-output-num	Whether to print the row numbers during table output. The value can be **true** (default) or **false**.	Use it directly in a command.
cli-endpoint	Custom domain name.	Use it directly in a command.

Parameter	Description	How to Use
cli-connect-timeout	Request connection timeout, in seconds. The default value is 5s, and the minimum value is 1s.	Configure it in a profile or use it directly in a command.
cli-read-timeout	I/O timeout, in seconds. The default value is 10s, and the minimum value is 1s.	Configure it in a profile or use it directly in a command.
cli-retry-count	Number of connection attempts. The value ranges from 0 to 5. The default value is **0**.	Configure it in a profile or use it directly in a command.
cli-security-token	Specifies a temporary token, which must be used together with a temporary AK/SK.	Configure it in a profile or use it directly in a command.
cli-lang	Language, which can be **cn** or **en**.	Configure it in a profile.
cli-offline	Specifies whether to use offline mode. The value can be **true** or **false**. The default value is **true**.	Configure it in a profile.
cli-skip-secure-verify	Specifies whether to skip HTTPS certificate verification (not recommended). The value can be **true** or **false** (default).	Configure it in a profile or use it directly in a command.
cli-agree-privacy-statement	Whether to agree to the privacy statement. The value can be **true** or **false** (default).	Configure it in a profile.
cli-warning	Whether to display warnings during command execution. The value can be **true** (default) or **false**.	Configure it in a profile.

The ways parameters listed in **the preceding table** can be used are described as follows:

- Configure it in a profile.

  The parameter can be used only after being configured in a profile by running **hcloud configure set --key1=value1 --key2=value2 ...** The profile name can be specified using **--cli-profile=***${profileName}*. KooCLI parses and uses the parameter values configured in the profile during the running process.

  If you use the parameter directly in a command, the following error message is displayed:

[USE_ERROR] Invalid parameter: *

- Use it directly in a command.

  The parameter can be used only in a command in the format of **--key1=value1 --key2=value2 ...**.

  If you configure the parameter in a profile, the following error message is displayed:

  [USE_ERROR] Invalid parameter: *

- Configure it in a profile or use it directly in a command.

  The parameter can be used after being configured in a profile by running **hcloud configure set --key1=value1 --key2=value2 ...**. Alternatively, it can be directly used in a command in the format of **--key1=value1 --key2=value2 ...** KooCLI preferentially uses the parameter value specified in the command. If the parameter is not specified in the command, its value in the current profile is used.

For details about the old KooCLI system parameters, see **Table 10-2**.

10.2 Why Am I Seeing a Message Indicating an Invalid Parameter?

Background

During a cloud service API call to manage and use your cloud service resources, KooCLI verifies the validity of your input parameters. If you enter an unsupported parameter in a command or pass a parameter that can be used only in a profile into a command, the following error message is displayed:

[USE_ERROR] Invalid parameter: *

Solution

1. Run the **hcloud <service> <operation> --help** command and check the parameters in **Params** of the command output. If the parameter does not exist, check either of the following reasons:
   a. Using the **online mode**: The API does not support this parameter. Modify the parameter based on the value of **Params** in the command output.
   b. Using the **offline mode**: The offline data package you used may not be the latest version, resulting in a parameter verification failure. In this case, run the **hcloud meta download** command to update the offline data package, and then run the command in **1** again. If the parameter still does not exist in the result, the API data is unavailable for KooCLI offline mode. Try again when the offline data package is updated, or **use the online mode** instead.
2. If you have entered system parameters (such as **cli-lang**) in the command for calling a cloud service API, an error message is displayed. The parameters can only be configured in a profile by running **hcloud configure set --key1=value1 --key2=value2 ...**
3. If the preceding error message is not displayed during the command execution, but the returned value of the called API indicates a parameter

error, the local cache files containing the API details may be modified. As a result, the parameter verification is inaccurate. In this case, run the **hcloud meta clear** command to clear the cache files and run the **hcloud <service> <operation> --help** command again to check whether the parameter is included in the parameter list of the target API.

10.3 Why Are Old and New KooCLI System Parameters (Such as region and cli-region) Available? Which Parameters Are Recommended?

Background

Among the KooCLI system parameters, some have two forms, for example, **--region** and **--cli-region**. The parameters without the prefix **cli-** are **old system parameters**. Those with the prefix **cli-** are **new system parameters**. This happens because some parameters of cloud service APIs may have the same names as old system parameters. Two parameters with the same name in a command may be used for different purposes. That is, one is used as a parameter of the target API, and the other is used as a system parameter. When checking the validity of parameters, the following error message may be displayed:

[USE_ERROR] Duplicate *. Change the key of the KooCLI system parameter to cli-*.

In addition, if the cloud service API contains a parameter (or a custom one) with the same name as an old system parameter, KooCLI cannot determine the function if the API parameter appears in a command. Therefore, when parsing the command, the CLI confirms the actual function of the parameter with you through interactive information to prevent errors. Example:

- The target API contains a parameter with the same name as the KooCLI system parameter %s (unknown location %s). Confirm whether this parameter is a KooCLI system parameter (a), a target API parameter (b), or both (c):
- You can define a parameter for the target API using the same name as the KooCLI system parameter %s (unknown location %s). Confirm whether this parameter is a KooCLI system parameter (a), a target API parameter (b), or both (c):

When you construct KooCLI commands, use the new system parameters to prevent errors or interactions caused by parameter conflicts.

NOTE

The new system parameters have been added to the new parameter list. The old system parameters can still be used but will not be upgraded.

Old System Parameters

The **following table** describes KooCLI's old system parameters and the corresponding new system parameters.

Table 10-2 KooCLI's old system parameters

Old Parameter	Description	New Parameter
region	Region.	cli-region
access-key	Access key ID required in AK/SK mode.	cli-access-key
secret-key	Secret access key required in AK/SK mode.	cli-secret-key
domain-id	Account ID.	cli-domain-id
project-id	Project ID.	cli-project-id
profile	Profile.	cli-profile
mode	Authentication mode (**AKSK** or **ecsAgency**).	cli-mode
jsonInput	Passes API parameters using a JSON file.	cli-jsonInput
output-cols	Specifies the fields to print during table output.	cli-output-cols
output-rows	Specifies the levels to print during table output.	cli-output-rows
output-num	Whether to print the row numbers during table output. The value can be **true** (default) or **false**.	cli-output-num
json-filter	Performs a JMESPath query on the output JSON result.	cli-json-filter
connect-timeout	Request connection timeout, in seconds. The default value is 5s, and the minimum value is 1s.	cli-connect-timeout
read-timeout	I/O timeout, in seconds. The default value is 10s, and the minimum value is 1s.	cli-read-timeout
retry-count	Specifies the number of connection attempts. The value ranges from 0 to 5. The default value is **0**.	cli-retry-count
security-token	Specifies a temporary token, which must be used together with a temporary AK/SK.	cli-security-token
lang	Language, which can be **cn** or **en**.	cli-lang

10.4 Why Am I Seeing a Message Indicating a Duplicate Parameter?

Background

KooCLI checks the validity of parameters during command execution. If a command contains duplicate parameters, different error messages will be displayed accordingly. For example:

1. [USE_ERROR] Duplicate parameter: *
2. [USE_ERROR] Duplicate *. Change the key of the KooCLI system parameter to cli-*.
3. [USE_ERROR] Duplicate *. Enter the API parameter using '--cli-jsonInput=xx'. For details, see...

Among the KooCLI system parameters, some have two forms, for example, **--region** and **--cli-region**. This happens because some parameters of cloud service APIs may have the same names as system parameters.

Solution

1. If the first error message is displayed, duplicate non-system parameters may exist in the command. Check whether the parameters are entered correctly. This error may also occur when the system parses the command content. If the parameter value contains special characters, enclose the value with double quotation marks ("") to prevent parsing errors.
2. If the second error message is displayed, a duplicate **old system parameter** exists in the command. Run the **hcloud <service> <operation> --help** command and compare the value of **Params** in the command output (that is, the parameter list of the current API) to check whether the parameter exists in the target API or whether a parameter with a custom name (that is, the parameter whose name is {*}) exists. If not, check whether the input is correct. If yes, use the two parameters with the same name for different purposes, one as a parameter of the target API, and the other as a system parameter. Replace the old system parameter in the command with the **new one** as prompted. If a command contains both the old and new forms of a system parameter, for example, **--cli-region=${regionValue1} --region=${regionValue2}**, KooCLI automatically identifies the usage of each parameter based on the parameter list of the current API.
   - If the target API has the **region** parameter or a parameter with a custom name, KooCLI automatically identifies **--cli-region** as a system parameter and uses its value to obtain details about the target API. KooCLI identifies **--region** as a parameter of the target API and uses its value to call the API.
   - If the target API does not have the **region** parameter or a parameter with a custom name, KooCLI automatically identifies **--cli-region** as a system parameter and uses its value to obtain details about the target API. KooCLI ignores **--region** passed to the command.

   When you construct KooCLI commands, **use the new system parameters** to prevent errors or interactions caused by parameter conflicts.

3. If the third error message is displayed, a duplicate new system parameter exists in the command. Run the **hcloud <service> <operation> --help** command and compare the value of **Params** in the command output (that is, the parameter list of the current API) to check whether the parameter exists in the target API or whether a parameter with a custom name (that is, the parameter whose name is {*}) exists. If not, check whether the input is correct. If yes (this conflict rarely occurs), write the API parameters in the command into the **cli-jsonInput** file and **call the command using this JSON file**.

10.5 How Do I Use cli-jsonInput?

Background

Tools such as the command prompt (**cmd.exe**) have restrictions on the allowed maximum string length. If there are too many parameters in a command or the parameter values are too long, the parameters entered may be incomplete due to the length limit. In this case, use **--cli-jsonInput=***${jsonFileName}* to pass cloud service API parameters through a JSON file. KooCLI parses and uses the parameters in the JSON file to call the target API.

Usage Description

For details about how to use **cli-jsonInput**, see **Passing API Parameters with JSON File**.

Precautions

- The JSON file passed by using the **--cli-jsonInput** option supports only API parameters of cloud services and does not support system parameters. If a parameter of the target API has the same name as **a new system parameter** or **an old system parameter**, the one written into the **jsonInput** file is identified as an API parameter by default.
- In the JSON file passed by using the **--cli-jsonInput** option, KooCLI obtains and parses the parameter value based on the key at the outermost layer of JSON. Currently, the supported keys include **path**, **query**, **body**, **formData**, **header**, and **cookie**. The content under other keys at the outermost layer of JSON will be ignored. If none of the keys at this layer is supported, the following error message is displayed:

  [USE_ERROR] The cli-jsonInput file contains invalid content. For details, see...
- When you use the **--cli-jsonInput** option to pass cloud service API parameters, all API parameters in the same position must be written into a JSON file or directly passed using a command. Otherwise, the parameters may not be completely parsed, and the following error message is displayed:

  [USE_ERROR] The following parameters are required: *
- The **--cli-jsonInput** option supports only JSON files with the .json extension. The maximum file size is 5 MB. When **--cli-jsonInput** is used, the JSON file format and the parameter types in the file are verified. If the JSON file has incorrect format, the following error message is displayed:

  [USE_ERROR]Failed to parse file for the cli-jsonInput parameter. Cause: File contains invalid parameters.

If the JSON file contains an unsupported parameter type, the following error message is displayed:

[USE_ERROR] Value type of * is not supported.

- When you use the **--cli-jsonInput** option to pass cloud service API parameters, parameter values cannot be custom parameters.

10.6 When Can I Use cli-jsonInput?

Use **cli-jsonInput** to pass the following API parameters so that they can be correctly parsed by KooCLI:

- API parameters with periods (.)
- API parameters with the same name in different positions
- API parameters with spaces or special characters
- API parameters that are too long

10.7 Why Am I Seeing a Message Indicating an Unsupported Parameter Position or Type?

Background

During a cloud service API call to manage and use your cloud service resources, KooCLI verifies the validity of your input parameters. KooCLI obtains details about all parameters of the API, including the parameter type and position in the request. If the locally cached metadata files are modified, KooCLI may not be able to correctly parse the API parameter details during the running process. The following error message will be displayed during verification:

- [CLI_ERROR] * is in an incorrect position: *
- [USE_ERROR] Unsupported parameter type: key=*, type=*

Solution

Run **hcloud meta clear** to clear the cached metadata files, and try again.

10.8 How Do I Leave a Body Parameter Empty for Cloud Service APIs?

KooCLI allows you to leave the body parameters of cloud service APIs empty at any level.

- A map parameter can be left empty as **{}** at the corresponding level.
- An array parameter can be left empty as **[]** at the corresponding level.

Take the **BatchStopServers** operation of ECS as an example. There are two parameters in the body: **os-stop.servers.[N].id** and **os-stop.type**.

```
hcloud ECS BatchStopServers --cli-region=ap-southeast-1 --help
```

```
KooCLI(Koo Command Line Interface) Version 3.2.8 Copyright(C) 2020-2023 www.huaweicloud.com

Service:
  ECS

Description:
This API is used to stop ECSs in a batch based on the specified ECS ID list. A maximum of 1000 ECSs can be
stopped at a time.

Method:
  POST

Params:
  --cli-region
    required    string  Region where the API can be called. If no region is specified in the command, cli-region
in the current profile is used.
  --os-stop.servers.[N].id
    required    string  body    ECS Instance ID. Format: --os-stop.servers.1.id=value1 ...
  --project_id
    required    string  path    Specifies the project ID. If no project ID is specified in the command, either the
parent project ID of the specified region in the authentication information or cli-project-id in the current
profile is used.
  --os-stop.type
    optional    string  body    Specifies an ECS stop type. The default value is SOFT. [SOFT|HARD]
    - SOFT: normal ECS stop (default)
    - HARD: forcible ECS stop
```

- No parameters left empty

  Pass the values of **os-stop.servers.[N].id** and **os-stop.type**, and run **--dryrun** to view the request body:

```
hcloud ECS BatchStopServers --cli-region=ap-southeast-1 --os-stop.servers.1.id="test" --os-
stop.type="SOFT" --dryrun
-------- The execution is eliminated in dry-run mode. Current request: --------
POST https://ecs.ap-southeast-1.myhuaweicloud.com/v1/0a152ab****************262d035e8/
cloudservers/action
Content-Type: application/json;charset=UTF-8
X-Project-Id: 0a152ab****************262d035e8
X-Sdk-Date: 20221116T121721Z
Authorization: ****

{
  "os-stop": {
    "servers": [
      {
        "id": "test"
      }
    ],
    "type": "SOFT"
  }
}
```

- Array parameter left empty

  Pass **--os-stop.servers="[]"** for the array parameter **os-stop.servers** of **os-stop.servers.[N].id** to leave **[N]** and the remaining part empty. Then run **--dryrun** to view the request body:

```
hcloud ECS BatchStopServers --cli-region=ap-southeast-1 --os-stop.servers="[]" --os-stop.type="SOFT"
--dryrun
-------- The execution is eliminated in dry-run mode. Current request: --------
POST https://ecs.ap-southeast-1.myhuaweicloud.com/v1/0a152ab****************262d035e8/
cloudservers/action
X-Project-Id: 0a152ab****************262d035e8
X-Sdk-Date: 20221116T122841Z
Authorization: ****
Content-Type: application/json;charset=UTF-8

{
  "os-stop": {
    "servers": [],
    "type": "SOFT"
  }
}
```

- Map parameter left empty

  Pass **--os-stop="{}"** for the common parent **os-stop** of **os-stop.servers.[N].id** and **os-stop.type** to leave the map parameter empty. Then run **--dryrun** to view the request body:

```
hcloud ECS BatchStopServers --cli-region=ap-southeast-1 --os-stop="{}" --dryrun
-------- The execution is eliminated in dry-run mode. Current request: --------
POST https://ecs.ap-southeast-1.myhuaweicloud.com/v1/0a152ab****************262d035e8/
cloudservers/action
Content-Type: application/json;charset=UTF-8
X-Project-Id: 0a152ab****************262d035e8
X-Sdk-Date: 20221117T013616Z
Authorization: ****

{
  "os-stop": {}
}
```

KooCLI checks the parameter values during command execution. It displays an error message if an empty value does not match the parameter type. For example, if the empty value **[]** for an array parameter is assigned to the map parameter **os-stop**, the following error message is displayed:

[USE_ERROR] The value of map parameter os-stop is invalid.

11 Interactive Mode and Autocomplete

11.1 How Do I Use Interactive Mode and Autocomplete?

To enable autocomplete in the bash environment, run the **hcloud auto-complete on** command. Pay attention to the following:

- If the prompted parameter name contains **[N]**, which indicates an index, replace it with a number. If the prompted parameter name contains **{*}**, which indicates a custom parameter name, replace it with a string that does not contain a period (.).
- If multiple directories of the same user contain KooCLI on the same server, the autocomplete function enabled for KooCLI in one of the directories will also take effect for the CLI in the other directories.

Note the following when using the interactive mode:

If the prompted parameter name contains **[N]**, which indicates an index, replace it with a number. If the prompted parameter name contains **{*}**, which indicates a custom parameter name, replace it with a string that does not contain a period (.).

12 Output Formats

12.1 What Output Formats Are Supported by KooCLI?

KooCLI supports three output formats: **json**, **table**, and **tsv**. By default, the output is in JSON format. Use the **--cli-output** parameter to specify any of the preceding output formats. Use the **--cli-query** parameter to pass a JMESPath expression and perform a JMESPath query on the JSON result to filter the required information. For details about the output, see **Specifying Output Format**. Learn how to **define a JMESPath expression**.

12.2 How Do I Define a JMESPath Expression?

12.2.1 How Do I Use JMESPath Expressions?

You can use JMESPath expressions as follows:

- Basic expression
  - Identifier

    The simplest JMESPath expression is an identifier, which specifies a key in a JSON object:

    ```
    {"a": "foo", "b": "bar", "c": "baz"}
    ```

    For the preceding JSON content, if the expression is **a**, the result **foo** is obtained.

If you specify a key that does not exist, KooCLI displays an error message and outputs the original JSON result.

– Subexpression

  Use a subexpression to return the nested value in a JSON object.

```
{"a": {"b": {"c": {"d": "value"}}}}
```

  For the preceding JSON content, if the expression is **a.b.c.d**, the result **value** is obtained.

  If you specify a key that does not exist, KooCLI displays an error message and outputs the original JSON result.

– Index expression

  Index expressions allow you to select a specific element in a list. Indexing is zero-based.

```
["a", "b", "c", "d", "e", "f"]
```

  For the preceding JSON content, if the expression is **[1]**, the result **b** is obtained.

  If you specify an index that is larger than the list, KooCLI displays an error message and outputs the original JSON result. You can also use negative indexing to index from the end of the list. **[-1]** indicates the last element in the list, and **[-2]** indicates the last but one element.

– You can combine identifiers, subexpressions, and index expressions to access JSON elements.

```
{"a": {
  "b": {
    "c": [
      {"d": [0, [1, 2]]},
      {"d": [3, 4]}
    ]
  }
}}
```

  For the preceding JSON content, if the expression is **a.b.c[0].d[1][0]**, the result **1** is obtained.

- Slice

  The general form of a slice is *[start:stop:step]*. By default, the step value is **1**. Therefore, the form can be *[start:stop]*. Slices allow you to select a contiguous subset of an array. In its simplest form, you can specify the starting index and the ending index. The ending index will not be included in the slice.

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

  For the preceding JSON content, if the expression is **[0:5]**, the following result is obtained:

```
[
  0,
  1,
  2,
  3,
  4
]
```

  This slice result contains the elements **0**, **1**, **2**, **3**, and **4**. The element at index **5** is not included.

  If the expression is **[5:10]**, the following result is obtained:

```
[
  5,
  6,
  7,
  8,
  9
]
```

The two examples above can be shortened. If the **start** or **stop** value is omitted, the array starts from the first element or stops from the last element by default. For example:

If the expression is **[:5]**, the following result is obtained:

```
[
  0,
  1,
  2,
  3,
  4
]
```

By default, the step value is **1**, which means to include every element in the range specified by the **start** and **stop** value. You can use the **step** value to skip over elements. For example, to select only the even elements from an array, use the expression **[::2]**. The following result is obtained:

```
[
  0,
  2,
  4,
  6,
  8
]
```

Also note that in this example, the **start** and **stop** values are omitted, meaning that **0** is used for the **start** value and **10** is used for the **stop** value. In this example, the expression **[::2]** is equivalent to **[0:10:2]**.

If the **step** value is negative, the slice is created in reverse order. For example, if the expression is **[::-1]**, the following result is obtained:

```
[
  9,
  8,
  7,
  6,
  5,
  4,
  3,
  2,
  1,
  0
]
```

- Projection

  Projections are one of the key features of JMESPath. It allows you to apply an expression to a collection of elements. There are five types of projection: list, slice, object, flatten, and filter projections.

  – List projection

    A wildcard expression creates a list projection, which is a projection over a JSON array.

    ```
    {
      "people": [
        {"first": "James", "last": "d"},
        {"first": "Jacob", "last": "e"},
        {"first": "Jayden", "last": "f"},
        {"missing": "different"}
      ],
      "foo": {"bar": "baz"}
    }
    ```

For the preceding JSON content, if the expression is **people[*].first**, the following result is obtained:

```
[
  "James",
  "Jacob",
  "Jayden"
]
```

In the above example, the **first** expression is applied to each element in the **people** array. The results are collected into a JSON array and returned as the result of the expression. For example, the expression **foo[*].bar.baz[0]** projects the **bar.baz[0]** expression to each element in the **foo** array.

Pay attention to the following when using projections:

- Projections are evaluated as two steps. The left hand side (LHS) creates a JSON array of initial values. The right hand side (RHS) of a projection is the expression to project for each element in the JSON array created by the LHS. Each projection type has slightly different semantics when evaluating the LHS and RHS.
- If the result of the expression projected onto an individual array element is **null**, then that value is omitted from the collected set of results.
- You can stop the projection using a pipe expression (discussed later).
- A list projection is only valid for a JSON array. If the LHS cannot create a JSON array of initial values, KooCLI displays an error message and outputs the original JSON result.

Note that **people[*].first** only included three elements, even though the **people** array has four elements. This is because the last element **{"missing": "different"}** evaluates to **null** when the expression **first** is applied, and **null** values are not added to the collected result array. If you try the expression **foo[*].bar**, KooCLI displays an error message and outputs the original JSON result, because the value associated with the **foo** key is a JSON object, not an array.

– Slice projection

Slice projections are almost identical to list projections, with the exception that the LHS is the result of evaluating the slice, which may not include all the elements in the original list:

```
{
  "people": [
    {"first": "James", "last": "d"},
    {"first": "Jacob", "last": "e"},
    {"first": "Jayden", "last": "f"},
    {"missing": "different"}
  ],
  "foo": {"bar": "baz"}
}
```

For the preceding JSON content, if the expression is **people[:2].first**, the following result is obtained:

```
[
  "James",
  "Jacob"
]
```

– Object projection

Whereas a list projection is defined for a JSON array, an object projection is defined for a JSON object. You can create an object projection using the ***** syntax. This will create a list of the values of the JSON object, and project the RHS of the projection onto the list of values.

```
{
  "ops": {
    "functionA": {"numArgs": 2},
    "functionB": {"numArgs": 3},
    "functionC": {"variadic": true}
  }
}
```

For the preceding JSON content, if the expression is **ops.*.numArgs**, the following result is obtained:

```
[
  2,
  3
]
```

The object projection can be divided into two parts. The LHS is **ops** and the RHS is **numArgs**. In the preceding example, ***** creates a JSON array of the values associated with the **ops** JSON object. The RHS of the projection, **numArgs**, is then applied to the JSON array, resulting in the final array of **[2, 3]**.

The following describes how to project an object:

i. The LHS is evaluated to create the initial array to be projected:

   evaluate(ops, inputData) -> [{"numArgs": 2}, {"numArgs": 3}, {"variadic": True}]

ii. Then the RHS is applied to each element in the array:

   evaluate(numArgs, {numArgs: 2}) -> 2

   evaluate(numArgs, {numArgs: 3}) -> 3

   evaluate(numArgs, {variadic: true}) -> null

iii. Any **null** values are not included in the final result, so the result of the entire expression is **[2, 3]**.

– Flatten projection

More than one projection can be used in a JMESPath expression. In the case of a list/object projection, the original data structure is preserved when a projection is created within another projection.

```
{
  "reservations": [
    {
      "instances": [
        {"state": "running"},
        {"state": "stopped"}
      ]
    },
    {
      "instances": [
        {"state": "terminated"},
        {"state": "running"}
      ]
    }
  ]
}
```

In the preceding JSON content, the expression **reservations[*].instances[*].state** indicates that the value of the toplevel key **reservations** is an array. For each element in the **reservations** array, project the **instances[*].state** expression. Within each element in the **reservations** array, there is an **instances** key whose value is also an array, and a **state** expression is projected for each element in the **instances** array. The following result is obtained:

```
[
  [
    "running",
    "stopped"
  ],
  [
    "terminated",
    "running"
  ]
]
```

The result is a nested list. The outer list is the projection of **reservations[*]**, and the inner list is the projection of **state** created from **instances[*]**.

What if you do not care which **reservations** the **instances** belongs to and you want a list of all the **state** values? That is, your expected result is as follows:

```
[
  "running",
  "stopped",
  "terminated",
  "running"
]
```

This is what a flatten projection solves. To get the expected result, you can use **[]** instead of **[*]** to flatten a list, that is, use **reservations[].instances[].state**.

Rules of thumb to use for the flatten operator **[]** are as follows:

- It flattens sublists into the parent list (not recursively, just one level).
- It creates a projection, so anything on the RHS of the flatten projection is projected onto the newly created flattened list.

You can also use **[]** on its own to flatten a list.

```
[
  [0, 1],
  2,
  [3],
  4,
  [5, [6, 7]]
]
```

For the preceding JSON content, if the expression is **[]**, the following result is obtained:

```
[
  0,
  1,
  2,
  3,
  4,
  5,
  [
    6,
    7
  ]
]
```

If you use **[][]** to flatten the result of the expression again, the result of **[0, 1, 2, 3, 4, 5, 6, 7]** is obtained.

– Filter projection

  Filter projections allow you to filter the LHS of the projection before evaluating the RHS of a projection.

```
{
  "machines": [
    {"name": "a", "state": "running"},
    {"name": "b", "state": "stopped"},
    {"name": "b", "state": "running"}
  ]
}
```

  For the preceding JSON content, if the expression is **machines[?state=='running'].name**, the following result is obtained:

```
[
  "a",
  "b"
]
```

  A filter expression is defined for an array and has the general form *LHS[?<Expression><Comparator><Expression>]RHS*. The following comparators are supported: ==, !=, <, <=, >, >=.

● Pipe expression

  Projection is an important concept in JMESPath. However, sometimes projection semantics are not what you want. A common scenario is when you want to operate the result of a projection rather than projecting an expression onto each element in the array. For example:

```
{
  "people": [
    {"first": "James", "last": "d"},
    {"first": "Jacob", "last": "e"},
    {"first": "Jayden", "last": "f"},
    {"missing": "different"}
  ],
  "foo": {"bar": "baz"}
}
```

  The expression **people[*].first** will give you an array containing the first names of everyone in the **people** array. What if you wanted the first element in that array? If you tried **people[*].first[0]** that you evaluate **first[0]** for each element in the **people** array, and because indexing is not defined for strings, the final result would be an empty array, **[]**. To obtain the desired result, you can use a pipe expression, *<Expression>* **|** *<expression>*. For the preceding JSON content, if the expression is **people[*].first | [0]**, the result is **James**.

  In the pipe expression above, the RHS of the list projection is **first**. When a pipe is encountered, the result up to that point is passed to the RHS of the pipe expression. The pipe expression is evaluated as:

  a. evaluate(people[*].first, inputData) -> ["James", "Jacob", "Jayden"]
  b. evaluate([0], ["James", "Jacob", "Jayden"]) -> "James"

● MultiSelect

  Multiselect expressions are classified into multiselect lists and multiselect hashes. Multiselect expressions allow you to create elements that do not exist in the JSON data. A multiselect list creates a list and a multiselect hash creates a JSON object.

– Multiselect list

```
{
  "people": [
    {
      "name": "a",
      "state": {"name": "up"}
    },
    {
      "name": "b",
      "state": {"name": "down"}
    },
    {
      "name": "c",
      "state": {"name": "up"}
    }
  ]
}
```

For the preceding JSON content, if the expression is **people[].[name,state.name]**, the following result is obtained:

```
[
  [
    "a",
    "up"
  ],
  [
    "b",
    "down"
  ],
  [
    "c",
    "up"
  ]
]
```

In the expression above, **[name,state.name]** is a multiselect list. It indicates that a list of two elements is created. The first element is the result of evaluating the **name** expression against the list element, and the second element is the result of evaluating **state.name**. Each list element will therefore create a two-element list, and the final result of the entire expression is a list of two-element lists.

Unlike a projection, the result of the expression is always included, even if the result is a null. If you change the above expression to **people[].[foo,bar]**, each two-element list will be **[null, null]**.

```
[
  [
    null,
    null
  ],
  [
    null,
    null
  ],
  [
    null,
    null
  ]
]
```

– Multiselect hash

A multiselect hash has the same basic idea of a multiselect list. The only difference is that a multiselect hash creates a hash instead of an array.

```
{
  "people": [
    {
      "name": "a",
      "state": {"name": "up"}
    },
    {
      "name": "b",
      "state": {"name": "down"}
    },
    {
      "name": "c",
      "state": {"name": "up"}
    }
  ]
}
```

For the preceding JSON content, if the expression is **people[].{Name:name,State:state.name}**, the following result is obtained:

```
[
  {
    "Name": "a",
    "State": "up"
  },
  {
    "Name": "b",
    "State": "down"
  },
  {
    "Name": "c",
    "State": "up"
  }
]
```

- Function

  JMESPath supports function expressions, for example:

```
{
  "people": [
    {
      "name": "b",
      "age": 30,
      "state": {"name": "up"}
    },
    {
      "name": "a",
      "age": 50,
      "state": {"name": "down"}
    },
    {
      "name": "c",
      "age": 40,
      "state": {"name": "up"}
    }
  ]
}
```

  For the preceding JSON content, if the expression is **length(people)**, the result is **3**.

  Functions can be used to transform and filter data in a powerful way. For details about the built-in functions, see **12.2.2 Which Built-in Functions Are Supported by JMESPath?**

  The following are some example functions.

  This example prints the name of the oldest person in the **people** array:

```
{
  "people": [
    {
      "name": "b",
      "age": 30
    },
    {
      "name": "a",
      "age": 50
    },
    {
      "name": "c",
      "age": 40
    }
  ]
}
```

For the preceding JSON content, if the expression is **max_by(people,&age).name**, the result is **a**.

Functions can also be combined with filter expressions. In the following example, the JMESPath expression finds all elements in **myarray** that contains the string **foo**.

```
{
  "myarray": [
    "foo",
    "foobar",
    "barfoo",
    "bar",
    "baz",
    "barbaz",
    "barfoobaz"
  ]
}
```

For the preceding JSON content, if the expression is **myarray[?contains(@,'foo')==`true`]**, the following result is obtained:

```
[
  "foo",
  "foobar",
  "barfoo",
  "barfoobaz"
]
```

The **@** character in the above example refers to the current element being evaluated in **myarray**. The expression **contains(@, `foo`)** will return **true** if the current element in the **myarray** array contains the string **foo**.

Pay attention to the following when using functions:

- Function arguments have types. If an argument for a function has the wrong type, KooCLI displays an error message and outputs the original JSON result. There are functions that can convert arguments (**to_string**, **to_number**) to their proper type.
- The number of function parameters is limited. If a function is called with the wrong number of arguments, KooCLI displays an error message and outputs the original JSON result.

12.2.2 Which Built-in Functions Are Supported by JMESPath?

The built-in functions of JMESPath support the following data types:

- number (integer and double-precision floating-point format in JSON)
- string
- boolean (true or false)
- array (an ordered sequence of values)
- object (an unordered collection of key-value pairs)

- expression (denoted by &expression)
- null

Built-in functions support different data types, as described in the following table. The special character **@** in function arguments indicates that the current result is passed to the function as an input parameter.

Table 12-1 Built-in functions supported by JMESPath expressions

Built-in Function	Input Data Type	Output Data Type	Description	Example
abs	number	number	Returns the absolute value of the provided argument.	• Expression: **abs(1)** Final result: **1** • Expression: **abs(-1)** Final result: **1**
avg	array[number]	number	Returns the average of the elements in the provided array.	Current result: **[10, 15, 20]** Expression: **avg(@)** Final result: **15**
ceil	number	number	Returns the next highest integer value by rounding up if necessary.	Expression: **ceil(`1.001`)** Final result: **2**
contains	array|string, any	boolean	Returns **true** if the first given argument contains the second one, or otherwise returns **false**.	• Expression: contains('foobar','foo') Final result: **true** • Current result: **["a", "b"]** Expression: **contains(@, 'a')** Final result: **true**
ends_with	string, string	boolean	Returns **true** if the first character string ends with the second one, or otherwise returns **false**.	Current result: **foobarbaz** Expression: **ends_with(@,'baz')** Final result: **true**

Built-in Function	Input Data Type	Output Data Type	Description	Example
floor	number	number	Returns the next lowest integer value by rounding down if necessary.	Expression: **floor(`1.001`)** Final result: **1**
join	string, array[string]	string	Returns all of the elements from the provided character string array joined using the given string argument as a separator.	Current result: **["a", "b"]** Expression: **join(',', @)** Final result: **"a, b"**
keys	object	array	Returns an array containing the keys of the provided JSON object. Because JSON hashes are inherited and unordered, the keys associated with the provided object are also inherited and unordered. Implementations are not required to return keys in any specific order.	● Current result: **{"foo": "baz", "bar": "bam"}** Expression: **keys(@)** The final result could be as follows: – ["foo", "bar"] – ["bar", "foo"] ● Current result: **{}** Expression: **keys(@)** Final result: **[]**

Built-in Function	Input Data Type	Output Data Type	Description	Example
length	string|array|object	number	Returns the length of the given argument using the following type rules: 1. **string**: Returns the number of characters in the string. 2. **array**: Returns the number of elements in the array. 3. **object**: Returns the number of key-value pairs in the object.	● Current result: **current** Expression: **length(@)** Final result: **7** ● Current result: **["a", "b", "c"]** Expression: **length(@)** Final result: **3** ● Current result: **{"foo": "bar", "baz": "bam"}** Expression: **length(@)** Final result: **2**

Built-in Function	Input Data Type	Output Data Type	Description	Example
map	expression->any->any, array[any]	array[any]	Applies the expression in the input argument to every element in the array, and returns the array of results. An element of length N will produce a return array of length N. Unlike a projection, **map()** will include the result of applying the expression for every element in the elements array, even if the result is null.	● Current result: **{"array": [{"foo": "a"}, {"foo": "b"}, {}, [], {"foo": "f"}]}** Expression: **map(&foo, array)** Final result: **["a", "b", null, null, "f"]** ● Current result: **[[1, 2, 3, [4]], [5, 6, 7, [8, 9]]]** Expression: **map(&[], @)** Final result: **[[1, 2, 3, 4], [5, 6, 7, 8, 9]]**
max	array[number]|array[string]	number	Returns the largest number in the provided array argument.	● Current result: **[10, 15]** Expression: **max(@)** Final result: **15** ● Current result: **["abc", "drb"]** Expression: **max(@)** Final result: **drb**

Built-in Function	Input Data Type	Output Data Type	Description	Example
max_by	array, expression->number|expression->string	any	Returns the maximum element in an array by using the expression as a comparison key.	Current result: **[{"name": "b", "age": 30, "age_str": "30"}, {"name": "a", "age": 50, "age_str": "50"}, {"name": "c", "age": 40, "age_str": "40"}]** For the preceding current result: ● Expression: **max_by(@, &age)** Final result: **{"age": 50, "age_str": "50", "name": "a"}** ● Expression: **max_by(@, &age).age** Final result: **50** ● Expression: **max_by(@, &to_number(age_str))** Final result: **{"age": 50, "age_str": "50", "name": "a"}**

Built-in Function	Input Data Type	Output Data Type	Description	Example
merge	[object [, object ...]]	object	Accepts one or more objects as arguments, and returns a single object with subsequent objects merged. The key-value pairs of each subsequent object are added to the preceding object. This function is used to combine multiple objects into one. You can think of this as the first object being the base object, and each subsequent argument being the overrides that are applied to the base object.	• Expression: **merge(`{"a": "b"}`, `{"c": "d"}`)** Final result: **{"a": "b", "c": "d"}** • Expression: **merge(`{"a": "b"}`, `{"a": "override"}`)** Final result: **{"a": "override"}** • Expression: **merge(`{"a": "x", "b": "y"}`, `{"b": "override", "c": "z"}`)** Final result: **{"a": "x", "b": "override", "c": "z"}**
min	array[number]|array[string]	number	Returns the smallest number in the provided array argument.	• Current result: **[10, 15]** Expression: **min(@)** Final result: **10** • Current result: **["a", "b"]** Expression: **min(@)** Final result: **"a"**

Built-in Function	Input Data Type	Output Data Type	Description	Example
min_by	array, expression->number|expression->string	any	Returns the smallest element in an array using the expression as a comparison key.	Current result: **{"people": [{"name": "b", "age": 30, "age_str": "30"}, {"name": "a", "age": 50, "age_str": "50"}, {"name": "c", "age": 40, "age_str": "40"}]}** For the preceding current result: ● Expression: **min_by(people, &age)** Final result: **{"age": 30, "age_str": "30", "name": "b"}** ● Expression: **min_by(people, &age).age** Final result: **30** ● Expression: **min_by(people, &to_number(age_str))** Final result: **{"age": 30, "age_str": "30", "name": "b"}**
not_null	[any [, any ...]]	any	This function accepts one or more arguments, and evaluates them in order until a non-null argument is encountered. If the values of all arguments as resolved as null, KooCLI displays an error message and outputs the original JSON result.	● Current result: **{"a": null, "b": null, "c": [], "d": "foo"}** Expression: **not_null(no_exist, a, b, c, d)** Final result: **[]** ● Current result: **{"a": null, "b": null, "c": [], "d": "foo"}** Expression: **not_null(a, b, \`null\`, d, c)** Final result: **"foo"** ● Current result: **{"a": null, "b": null, "c": [], "d": "foo"}** Expression: **not_null(a, b)** Final result: **null**

Built-in Function	**Input Data Type**	**Output Data Type**	**Description**	**Example**
reverse	string|array	string|array	Reverses the order of the input argument.	● Current result: **[0, 1, 2, 3, 4]** Expression: **reverse(@)** Final result: **[4, 3, 2, 1, 0]** ● Current result: **[]** Expression: **reverse(@)** Final result: **[]** ● Current result: **["a", "b", "c"]** Expression: **reverse(@)** Final result: **["c", "b", "a"]** ● Current result: **"abcd"** Expression: **reverse(@)** Final result: **"dcba"**
sort	array[number]|array[string]	array	Accepts an array argument and returns the sorted elements as an array. The array must be a list of strings or numbers. Strings are sorted based on a dictionary.	● Current result: **["b", "a", "c"]** Expression: **sort(@)** Final result: **["a", "b", "c"]** ● Current result: **[1, 4, 2]** Expression: **sort(@)** Final result: **[1, 2, 4]**

Built-in Function	Input Data Type	Output Data Type	Description	Example
sort_by	array, expression->number|expression->string	-	Sorts an array using an expression as the sort key. For each element in the array of elements, the expression is applied and the result value is used as the key for sorting the elements. sort_by follows the same sorting logic as the sort function.	Current result: **{"people": [{"name": "b", "age": 30, "age_str": "30"}, {"name": "a", "age": 50, "age_str": "50"}, {"name": "c", "age": 40, "age_str": "40"}]}** For the preceding current result: ● Expression: **sort_by(people, &age)[].age** Final result: **[30, 40, 50]** ● Expression: **sort_by(people, &age)[0]** Final result: **{"age": 30, "age_str": "30", "name": "b"}** ● Expression: **sort_by(people, &to_number(age_str))[1]** Final result: **{"age": 40, "age_str": "40", "name": c}**
starts_with	string, string	boolean	Returns **true** if the first argument starts with the second one, or otherwise returns **false**.	● Current result: **foobarbaz** Expression: **starts_with(@, 'foo')** Final result: **true** ● Current result: **foobarbaz** Expression: **starts_with(@,'baz')** Final result: **false** ● Current result: **foobarbaz** Expression: **starts_with(@, 'f')** Final result: **true**
sum	array[number]	number	Returns the sum of the provided array argument. An empty array will produce a return value of **0**.	● Current result: **[10, 15]** Expression: **sum(@)** Final result: **25** ● Current result: **[]** Expression: **sum(@)** Final result: **0**

Built-in Function	Input Data Type	Output Data Type	Description	Example
to_array	any	array	**array**: Returns the passed value. **number|string|object|boolean**: Returns a one-element array containing the passed argument.	● Expression: **to_array(`[1, 2]`)** Final result: **[1, 2]** ● Expression: **to_array('string')** Final result: **["string"]** ● Expression: **to_array(`0`)** Final result: **[0]** ● Expression: **to_array(`true`)** Final result: **[true]** ● Expression: **to_array(`{"foo": "bar"}`)** Final result: **[{"foo": "bar"}]**
to_string	any	string	**string**: Returns the passed value. **number|array|object|boolean**: The JSON encoded value of the object.	● Expression: **to_string(`2`)** Final result: **2** ● Expression: **to_string(`[]`)** Final result: **"[]"** ● Expression: **to_string(false)** Final result: **"null"**
to_number	any	number	**string**: Returns the parsed number. **number**: Returns the passed value. **array|object|boolean|null**: KooCLI displays an error message and outputs the original JSON result.	● Expression: **to_number(`2.3`)** Final result: **2.3** ● Expression: **to_number(`2`)** Final result: **2**

Built-in Function	Input Data Type	Output Data Type	Description	Example
type	array| object| string| number| boolean| null	string	Returns the data type of the given argument as a string value. The return value must be one of the following: ● "number" ● "string" ● "boolean" ● "array" ● "object" ● "null"	● Expression: **type('foo')** Final result: **"string"** ● Expression: **type(\`true\`)** Final result: **"boolean"** ● Expression: **type(\`null\`)** Final result: **"null"** ● Expression: **type(\`123\`)** Final result: **number** ● Expression: **type(\`123.05\`)** Final result: **number** ● Expression: **type(\`[1,2]\`)** Final result: **"array"** ● Current result: **{"abc": "123"}** Expression: **type(@)** Final result: **"object"**
values	object	array	Returns an array of values of the provided JSON object. Because JSON hashes are inherited and unordered, the values associated with the provided object are also inherited and unordered. Implementations are not required to return values of the JSON object in any specific order.	Current result: **{"a": "first", "b": "second", "c": "third"}** Expression: **values(@)** The final result could be as follows: ● ["first", "second", "third"] ● ["first", "third", "second"] ● ["second", "first", "third"] ● ["second", "third", "first"] ● ["third", "first", "second"] ● ["third", "second", "first"]

12.3 Which KooCLI System Parameters Are Related to Data Output? Which Ones Are Recommended?

The following table lists the KooCLI system parameters related to data output.

Table 12-2 KooCLI system parameters related to data output

Category	Parameter	Description
New output parameters	cli-output, cli-query, cli-output-num	● cli-output Response data output format. The options are as follows: – json – table – tsv ● cli-query JMESPath for filtering response data. ● cli-output-num Whether to print the row numbers during table output. The value can be **true** or **false**.
Old output parameters	cli-output-rows, cli-output-cols, cli-output-num, cli-json-filter	● cli-output-rows Levels to print during table output. ● cli-output-cols Fields to print during table output. ● cli-output-num Whether to print the row numbers during table output. The value can be **true** or **false**. ● cli-json-filter Performs a JMESPath query on the output JSON result.

Compared with the old output parameters, the new output parameters support TSV format in addition to table and JSON and are unified to facilitate user operations.

Functions related to output formats will be continuously optimized based on the new output parameters. The old output parameters can still be used but will not be upgraded. **You are advised to use the new output parameters** when constructing commands.

12.4 How Do I Use cli-output, cli-query, and cli-output-num?

For details about how to use the new output parameters, see **Specifying Output Format**.

In a command, use **--cli-query** to pass a **JMESPath** expression and perform a **JMESPath** query on the result to extract key information in the original returned result. Use **--cli-output** to specify the response data output format. Use **--cli-output-num** to specify whether to print the row numbers during table output.

When you use the preceding parameters, note that:

- Use only **--cli-output** in a command to specify the output format. If you use only **--cli-query** in a command, the default output format is JSON.
- When using **--cli-query**, enclose the value with double quotation marks ("") to avoid parsing errors when the system processes commands.
- When using **--cli-output-num** to specify whether to print the row numbers, set **--cli-output** to **table**.
- In the same command, if **--cli-output** has been used and old system parameters such as **--cli-output-rows** and **--cli-json-filter** are specified, the value of **--cli-output** is preferentially used as the target output format.

12.5 How Do I Use cli-output-rows, cli-output-cols, and cli-output-num? What Are the Precautions?

12.5.1 How Do I Use cli-output-rows, cli-output-cols, and cli-output-num?

By default, results are returned in JSON format when you call cloud service APIs using KooCLI. KooCLI supports the **--cli-output-rows**, **--cli-output-cols**, and **--cli-output-num** parameters to output data in table format. These parameters facilitate the extraction of key information in the calling result.

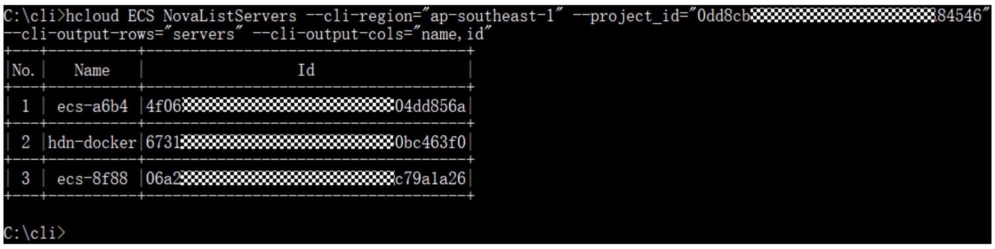
By default, the original calling result is output in JSON format:

```
hcloud ECS NovaListServers --cli-region="ap-southeast-1" --project_id="0dd8cb****************19b5a84546"
{
  "servers": [
    {
      "name": "ecs-a6b4",
      "links": [
        {
          "rel": "self",
          "href": "https://ecs.ap-southeast-1.myhuaweicloud.com/v2.1/0dd8cb****************19b5a84546/servers/
4f06****-****-****-****-****04dd856a"
        },
        {
```

```
        "rel": "bookmark",
        "href": "https://ecs.ap-southeast-1.myhuaweicloud.com/0dd8cb****************19b5a84546/servers/
4f06****-****-****-****-****04dd856a"
      }
    ],
    "id": "4f06****-****-****-****-****04dd856a"
  },
  {
    "name": "hdn-docker",
    "links": [
      {
        "rel": "self",
        "href": "https://ecs.ap-southeast-1.myhuaweicloud.com/v2.1/0dd8cb****************19b5a84546/servers/
6731****-****-****-****-****0bc463f0"
      },
      {
        "rel": "bookmark",
        "href": "https://ecs.ap-southeast-1.myhuaweicloud.com/0dd8cb****************19b5a84546/servers/
6731****-****-****-****-****0bc463f0"
      }
    ],
    "id": "6731****-****-****-****-****0bc463f0"
  },
  {
    "name": "ecs-8f88",
    "links": [
      {
        "rel": "self",
        "href": "https://ecs.ap-southeast-1.myhuaweicloud.com/v2.1/0dd8cb****************19b5a84546/servers/
06a2****-****-****-****-****c79a1a26"
      },
      {
        "rel": "bookmark",
        "href": "https://ecs.ap-southeast-1.myhuaweicloud.com/0dd8cb****************19b5a84546/servers/
06a2****-****-****-****-****c79a1a26"
      }
    ],
    "id": "06a2****-****-****-****-****c79a1a26"
  }
 ]
}
```

During table output, **--cli-output-rows** specifies a JSON structure level, that is, the data source of the table, **--cli-output-cols** specifies the column names of the table, which must correspond to the fields in the JSON structure, and **--cli-output-num** specifies whether to print the row numbers of the table (the default value is **true**).

hcloud ECS NovaListServers --cli-region="ap-southeast-1" --project_id="0dd8cb******************b5a84546" --cli-output-rows="servers" --cli-output-cols="name,id"

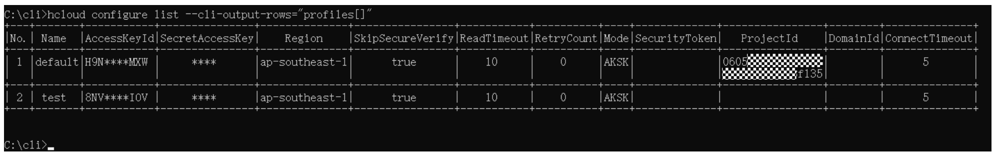


```
C:\cli>hcloud ECS NovaListServers --cli-region="ap-southeast-1" --project_id="0dd8cb[redacted]84546"
--cli-output-rows="servers" --cli-output-cols="name,id"
+---+----------+------------------------------------+
|No.|   Name   |                 Id                 |
+---+----------+------------------------------------+
| 1 | ecs-a6b4 |4f06[redacted]04dd856a|
+---+----------+------------------------------------+
| 2 |hdn-docker|6731[redacted]0bc463f0|
+---+----------+------------------------------------+
| 3 | ecs-8f88 |06a2[redacted]c79a1a26|
+---+----------+------------------------------------+

C:\cli>
```

The **--cli-output-rows**, **--cli-output-cols**, and **--cli-output-num** parameters can also be used in system commands. For example:

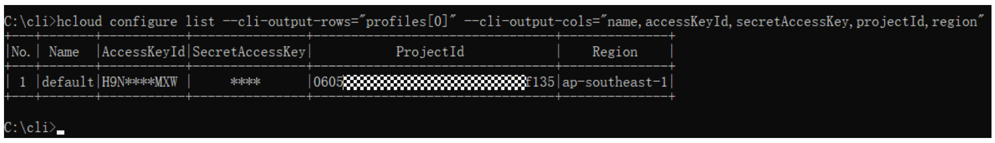
hcloud configure list --cli-output-rows="profiles[]"



```
C:\cli>hcloud configure list --cli-output-rows="profiles[]"
+---+-------+-----------+---------------+--------------+----------------+-----------+----------+----+-------------+-----------------+--------+--------------+
|No.| Name  |AccessKeyId|SecretAccessKey|    Region    |SkipSecureVerify|ReadTimeout|RetryCount|Mode|SecurityToken|    ProjectId    |DomainId|ConnectTimeout|
+---+-------+-----------+---------------+--------------+----------------+-----------+----------+----+-------------+-----------------+--------+--------------+
| 1 |default|H9N****MXW |     ****      |ap-southeast-1|      true      |    10     |    0     |AKSK|             |0605[redacted]   |        |      5       |
|   |       |           |               |              |                |           |          |    |             |            f135 |        |              |
+---+-------+-----------+---------------+--------------+----------------+-----------+----------+----+-------------+-----------------+--------+--------------+
| 2 | test  |8NV****IOV |     ****      |ap-southeast-1|      true      |    10     |    0     |AKSK|             |                 |        |      5       |
+---+-------+-----------+---------------+--------------+----------------+-----------+----------+----+-------------+-----------------+--------+--------------+

C:\cli>
```

hcloud configure list --cli-output-rows="profiles[0]" --cli-output-cols="name,accessKeyId,secretAccessKey,projectId,region"



```
C:\cli>hcloud configure list --cli-output-rows="profiles[0]" --cli-output-cols="name,accessKeyId,secretAccessKey,projectId,region"
+---+-------+-----------+---------------+--------------------------------+--------------+
|No.| Name  |AccessKeyId|SecretAccessKey|           ProjectId            |    Region    |
+---+-------+-----------+---------------+--------------------------------+--------------+
| 1 |default|H9N****MXW |     ****      |0605[redacted]f135              |ap-southeast-1|
+---+-------+-----------+---------------+--------------------------------+--------------+

C:\cli>
```

For details about how to use **--cli-output-rows**, **--cli-output-cols**, and **--cli-output-num**, see **12.5.2 What Are the Precautions for Using cli-output-rows, cli-output-cols, and cli-output-num?**

12.5.2 What Are the Precautions for Using cli-output-rows, cli-output-cols, and cli-output-num?

If **--cli-output-rows**, **--cli-output-cols**, and **--cli-output-num** are used in a command, the command output is in table format. Table output helps you extract key information from return values. The parameters are described as follows:

- **--cli-output-cols**: the fields to print during table output.
- **--cli-output-rows**: the levels to print during table output. For example, if you want to convert a JSON structure to a table, set this parameter to the name of the structure.
- **--cli-output-num**: whether to print the row numbers during table output. The value can be **true** (default) or **false**.

For details, see **12.5.1 How Do I Use cli-output-rows, cli-output-cols, and cli-output-num?**

Note the following when using the preceding parameters for table output:

- **--cli-output-cols** and **--cli-output-rows** can be used separately or together.
  - Use only **--cli-output-rows**.

    The **--cli-output-rows** option is separately used in a command to pass the name of a JSON structure in the calling result. The levels must be separated by periods (.). The content of the target JSON structure must be an array. KooCLI will display the structure content in a table. For example, if you run the **hcloud configure list --cli-output-rows=profiles** command, all profile information will be displayed in a table. If the JSON structure specified in **--cli-output-rows** is not an array, the following error message is displayed:

    [CLI_ERROR] Table output error. The cli-output-cols parameter is required.
  - Use only **--cli-output-cols**.

    The **--cli-output-cols** option is used to pass the root element fields of the JSON structure of the calling result. The fields must be separated by commas (,). For example, if you run the **hcloud configure show --cli-profile=*${profileName}* --cli-output-cols=accessKeyId** command, the access key ID in the specified profile will be displayed in a table. When **--cli-output-cols** is used separately, only the root element fields of the JSON structure can be specified. Otherwise, the following error message is displayed:

[USE_ERROR] In parameter cli-output-cols, * is null.

– Use both **--cli-output-cols** and **--cli-output-rows**.

The **--cli-output-rows** option is used to specify the levels to print, and the **--cli-output-cols** option is used to specify the fields to print at the levels. For example, if you run the **hcloud configure list --cli-output-rows=profiles --cli-output-cols=accessKeyId** command, the access key IDs of all profiles will be displayed in a table.

You can set **--cli-output-rows** to **[n]** or **[m:n]** to specify the index of the array element to be printed. Specify **[n]** to print the value of index n, and specify **[m:n]** to print the values of indexes m to (n – 1). For example, if you run the **hcloud configure list --cli-output-rows=profiles[0:2] --cli-output-cols=accessKeyId** command, the access key IDs of the profiles whose indexes are 0 and 1 in the profile array are displayed in a table. Pay attention to the following:

- If the index value of the array in **--cli-output-rows** is **[m:n]** and the value of **n** exceeds the array length limit, data within the maximum index will be printed.
- If the index value of the array in **--cli-output-rows** is **[n]** and the value of **n** exceeds the array length limit, an error message is displayed indicating that the array index is out of range:

  [USE_ERROR] The cli-output-rows parameter contains an incorrect field (*). The array index is out of range and the array length is *. Current index: *.
- When both **--cli-output-cols** and **--cli-output-rows** are used, the parameters in **--cli-output-rows** do not need to be of the array type. You only need to specify the level.

- When **--cli-output-num** is used separately, the output will not be displayed in a table.
- In the same command, **--cli-output-rows**, **--cli-output-cols**, and **--cli-output-num** cannot be used together with **--cli-json-filter**. Otherwise, an error occurs because the output format cannot be determined.

12.6 How Do I Use cli-json-filter? What Are the Precautions?

12.6.1 How Do I Use cli-json-filter?

By default, results are returned in JSON format when you call cloud service APIs using KooCLI. KooCLI supports the **--cli-json-filter** parameter to perform JMESPath query on JSON results. This option facilitates the extraction of key information in the results. For example:

By default, the original calling result is output in JSON format:

```
hcloud ECS NovaListServers --cli-region="ap-southeast-1" --project_id="0dd8cb****************19b5a84546"
{
  "servers": [
    {
      "name": "ecs-a6b4",
      "links": [
        {
          "rel": "self",
          "href": "https://ecs.ap-southeast-1.myhuaweicloud.com/v2.1/0dd8cb****************19b5a84546/servers/
4f06****-****-****-****-****04dd856a"
        },
        {
          "rel": "bookmark",
          "href": "https://ecs.ap-southeast-1.myhuaweicloud.com/0dd8cb****************19b5a84546/servers/
4f06****-****-****-****-****04dd856a"
        }
      ],
      "id": "4f06****-****-****-****-****04dd856a"
    },
    {
      "name": "hdn-docker",
      "links": [
        {
          "rel": "self",
          "href": "https://ecs.ap-southeast-1.myhuaweicloud.com/v2.1/0dd8cb****************19b5a84546/servers/
6731****-****-****-****-****0bc463f0"
        },
        {
          "rel": "bookmark",
          "href": "https://ecs.ap-southeast-1.myhuaweicloud.com/0dd8cb****************19b5a84546/servers/
6731****-****-****-****-****0bc463f0"
        }
      ],
      "id": "6731****-****-****-****-****0bc463f0"
    },
    {
      "name": "ecs-8f88",
      "links": [
        {
          "rel": "self",
          "href": "https://ecs.ap-southeast-1.myhuaweicloud.com/v2.1/0dd8cb****************19b5a84546/servers/
06a2****-****-****-****-****c79a1a26"
        },
        {
          "rel": "bookmark",
          "href": "https://ecs.ap-southeast-1.myhuaweicloud.com/0dd8cb****************19b5a84546/servers/
06a2****-****-****-****-****c79a1a26"
        }
      ],
      "id": "06a2****-****-****-****-****c79a1a26"
    }
  ]
}
```

Use **--cli-json-filter** to perform JMESPath query on the original JSON result, obtain the **id** and **name** of each **servers** element, and rename them **EcsID** and **EcsName**, respectively. The following is an example:

```
hcloud ECS NovaListServers --cli-region="ap-southeast-1" --project_id="0dd8cb****************19b5a84546" --
cli-json-filter="servers[].{EcsID:id,EcsName:name}"
[
  {
    "EcsID": "4f06****-****-****-****-****04dd856a",
    "EcsName": "ecs-a6b4"
  },
  {
    "EcsID": "6731****-****-****-****-****0bc463f0",
    "EcsName": "hdn-docker"
  },
  {
    "EcsID": "06a2****-****-****-****-****c79a1a26",
    "EcsName": "ecs-8f88"
  }
]
```

--cli-json-filter can also be used in system commands. For example, to query all **custom** parameters of the profile **test**, run the following commands:

```
hcloud configure list --cli-custom=true --cli-json-filter="profiles[?name=='test'].custom"
[
 {
   "password": {
     "isEncrypted": true,
     "value": "****"
   },
   "projectId": {
     "isEncrypted": false,
     "value": "068100000000000000000000f89d2e"
   }
 }
]
```

For more information about how to use **--cli-json-filter**, see **What Are the Precautions for Using cli-json-filter?**

12.6.2 What Are the Precautions for Using cli-json-filter?

KooCLI supports the **--cli-json-filter** option to pass a JMESPath expression and perform a JMESPath query on the JSON result. This option facilitates the extraction of key information in the results. Note the following when using **--cli-json-filter**:

- If **--cli-json-filter** is used in the command, the calling result will be output in JSON format.
- In the same command, **--cli-json-filter** cannot be used together with **--cli-output-rows**, **--cli-output-cols**, or **--cli-output-num**. Otherwise, an error occurs because the output format cannot be determined.

For details about using **cli-json-filter** to define a JMESPath expression, see the following:

- **12.2.1 How Do I Use JMESPath Expressions?**
- **12.2.2 Which Built-in Functions Are Supported by JMESPath?**

13 Other

13.1 How Do I Use KooCLI in Non-configuration Mode?

When using KooCLI in non-configuration mode, you do not need to pass your authentication information through a profile. Instead, directly pass your authentication parameters in commands. This mode enables you to use the CLI conveniently without adding any profiles. For details, see **Using KooCLI in Non-configuration Mode**.

Note the following when using the KooCLI in this mode:

- Using an AK/SK in non-configuration mode
  - Access key (permanent AK/SK)
    - When cloud service APIs are called using a permanent AK/SK, pass both the access key ID (**cli-access-key**) and secret access key (**cli-secret-key**) in commands for authentication.
    - If a global service is to be accessed, the ID (**cli-domain-id**) of the account used to create the IAM user is also required for authentication. If the ID is not passed to the command, KooCLI automatically obtains it based on the user authentication information. However, if the **cli-access-key** or **cli-secret-key** parameter is missing or the **cli-domain-id** parameter fails to be automatically obtained, the following error message is displayed:
      - [USE_ERROR] Parameters cli-access-key,cli-secret-key must be specified at the same time.
      - [USE_ERROR] cli-domain-id is required for access to global services using AK/SK. Add this parameter or run `hcloud configure set` to configure it.

        - If both an AK/SK and a profile (**cli-profile**) are specified in a command, the AK/SK is preferentially used for authentication.
    - Temporary security credentials (temporary AK/SK and SecurityToken)
        - Using a temporary AK/SK and SecurityToken to call cloud service APIs through KooCLI is similar to using a permanent AK/SK. If both an AK/SK (**cli-access-key**/**cli-secret-key**) and SecurityToken (**cli-security-token**) are passed in the command, the AK/SK is considered temporary.
        - If a global service is to be accessed, the ID (**cli-domain-id**) of the account used to create the IAM user is also required for authentication. If the ID is not passed to the command, KooCLI automatically obtains it based on the user authentication information. However, if the **cli-access-key** or **cli-secret-key** parameter is missing or the **cli-domain-id** parameter fails to be automatically obtained, the following error message is displayed:
            - [USE_ERROR] Parameters cli-access-key,cli-secret-key must be specified at the same time.
            - [USE_ERROR] cli-domain-id is required for access to global services using AK/SK. Add this parameter or run `hcloud configure set` to configure it.
        - If a temporary AK/SK, SecurityToken, and profile (**cli-profile**) are specified in a command, the AK/SK and SecurityToken are preferentially used for authentication.
- Using an ECS agency in non-configuration mode
    - This authentication mode applies only when you use KooCLI on an ECS.
    - To use this mode, create a cloud service agency to delegate ECS to use the CLI on the IAM console, and add the agency in the **Management Information** > **Agency** area of the ECS details page. For details, see **Cloud Service Delegation**.

13.2 Should I Enclose a Service Name, Operation, and Parameter Value in Quotation Marks in a Command?

This depends on the value.

Generally, the service name, operation, and parameter value in a command do not need to be enclosed in double quotation marks. If a value contains special characters, spaces, or symbols that need to be escaped, use double quotation marks to enclose the value.

Obtain CLI examples on API Explorer rather than manually entering parameters in commands.

13.3 What Are the Application Scenarios of Online/ Offline Modes?

- Viewing/Switching the current mode

  KooCLI can stay in online/offline modes. By default, it is offline. After configuring a profile, you can run **hcloud configure list --cli-query=offline** to check whether the offline mode is used.

  – To switch to offline mode, run the **hcloud configure set --cli-offline=true** command.

  – To switch to online mode, run the **hcloud configure set --cli-offline=false** command.

- Offline mode scenario

  Download the latest offline metadata of KooCLI to a local directory. The metadata cache files have unlimited validity. When you run KooCLI commands, the file content will be read for command verification and parsing. In this mode, local metadata cache files will not be automatically updated, and parameter verification for existing commands will not be affected. This ensures that the built KooCLI commands are always available. Use this mode if you build scripts with KooCLI commands and periodically execute the scripts to manage cloud services and resources.

- Online mode scenario

  KooCLI obtains metadata and caches it locally during command execution. The metadata cache files have limited validity. If a file has expired, KooCLI updates it and then uses it for command verification and parsing. In this mode, only the metadata related to your executed commands is saved, and you can call APIs of new cloud services through KooCLI. Use this mode if you need to run any KooCLI commands immediately to manage cloud services and resources.

13.4 How Do I Uninstall KooCLI?

KooCLI can be used without installation. To uninstall the CLI, delete it and the related local cache files by performing the following steps:

1. Run the **hcloud auto-complete off** command to turn off autocomplete.
2. Delete the cache files, configuration files, and log files.
   – Linux: **/home/***{Current username}***/.hcloud/**
   – Windows: **C:\Users\***{Current username}***\\.hcloud\**
   – macOS: **/Users/***{Current username}***/.hcloud/**
3. Delete KooCLI.