ParallelGuard Bash/CLI SDK Manual

Version 1.2

There are SDKs in various languages available for ParallelGuard to help programs call the ParallelGuard RESTful APIs.

Bash SDK helps Bash scripts or command line tools to invoke the ParallelGuard APIs.

Required tools

The Bash SDK requires the following tools to work.

```
mktemp
curl
```

Please make sure these tools are installed before using the SDK.

About GenSig

The SDK contains a pre-build **`GenSig`** tool used by the CLI SDK to generate certificate for the PUT requests. If the version from the SDK release does not work in your environment, you may try to re-build it manaully by following steps in the SDK directory using Go (make sure you have Go development environment installed correctly).

```
# First get the ParallelGuard Go SDK library
get -u github.com/digital-transaction/parallelguard-sdk-go
# Build the GenSig, make sure you are in the CLI SDK directory
go build GenSig.go
```

How to use the Bash SDK

In your program, souce the **`parallelguard.sh`** file so that the functions are available.

```
source /path/to/parallelguard.sh
```

SDK functions

The SDK provides following functions.

`parallelguard_get`

Get the value of a key from ParallelGuard.

Arguments:

```
$1: service_end_point
$2: the key
$3: path of the file the value or the error message
    will be saved to, or '-' for STDOUT
```

Return code:

```
0: successful
1: failed to send request to the end point
2: HTTP status code is not 200
```

Check the API specification document for more details.

`parallelguard_put`

Put value for a key to ParallelGuard.

Arguments:

```
$1: service_end_point
$2: the key
$3: path to a file containing the value
$4: the private key string of that key space
$5: path of the file the response message will be saved to,
    or '-' for STDOUT
```

Return code:

```
0: successful
1: failed to send request to the end point
2: HTTP status code is not 200
```

Check the API specification document for more details.

`parallelguard_list_versions`

Get the versions list of a key from ParallelGuard.

Arguments:

```
$1: service_end_point
$2: the key
$3: path of the file the value or the error message
    will be saved to, or '-' for STDOUT
$4: withtimestamp, a string, set to 'false' or 'true'
    default is 'false'
$5: maxcount, a number in string format
    default is '100'
$6: olderthan, a version_id, tell API where to stop fetching version_id
```

Return code:

```
0: successful
1: failed to send request to the end point
2: HTTP status code is not 200
```

Check the API specification document for more details.

`parallelguard_get_version`

Get the value of a key for specific version from ParallelGuard.

Arguments:

```
$1: service_end_point
$2: $key
$3  $verison
$4: path of the file the value or the error message
    will be saved to, or '-' for STDOUT
```

Return code:

```
0: successful
1: failed to send request to the end point
2: HTTP status code is not 200
```

Check the API specification document for more details.

Example usage and command line tools

There are tools **`parallelguard-put.sh`** , **`parallelguard-get.sh`**, **`parallelguard-list-versions.sh`** and **`parallelguard-get-version.sh`** which are based on the Bash SDK.

These tools can be invoked directly. Usages are as follows.

`parallelguard-get.sh`:

```
Usage: parallelguard-get.sh end_point_url key value_file_path
```

`parallelguard-put.sh`:

```
  Usage: parallelguard-put.sh end_point_url key value_file_path passphrase
  output_file_path
```

parallelguard-list-versions.sh:

```
  Usage: parallelguard-list-versions.sh end_point_url key version_list_file_path
```

parallelguard-get-version.sh:

```
  Usage: parallelguard-get-version.sh end_point_url key version value_file_path
```