PEG REST API on PrivaceraCloud

The PEG API endpoint is obtained using the **Copy Url** link in **Settings > Api Key**.

In the examples here, we call this endpoint `<cloud_peg_api_endpoint>` .

The PEG REST API consists of the following endpoints:

- `/protect` : encrypts the data
- `/unprotect` : decrypts the data

REST API prerequisites

API Key

For the REST API requests `protect` and `/unprotect` , you need an API key. See API Key on PrivaceraCloud[https://privacera.com/docs/en/api-key-on-privaceracloud.html].

Scheme policy required for protect and unprotect API endpoints

For the REST API endpoints `/protect` and `/unprotect` , you must create a scheme policy that grants these permissions to the user. See Scheme policies[https://privacera.com/docs/en/scheme-policies.html].

Anatomy of a PEG API endpoint on PrivaceraCloud

This example of the `/protect` endpoint illustrates some common fields of the PEG REST API on PrivaceraCloud. The example is split across separate lines for clarity but in actual use is a single line.

```
curl -il
--request POST https://<cloud_peg_api_endpoint>/api/<api-key>/api/peg/public/protect
-u <service_user>:<password>
--header "Accept: application/json"
--header 'Content-Type: application/json'

--data-raw '{"schemelist":["<encryption_scheme>,..."], \
     "datalist":[["<data_to_encrypt>",...]], \
     "maskSchemelist": ["<masking_scheme>",...], \
     "maskDatalist": [[data_to_mask,...], \
     "user":"<application_user>"}'
```

Line	Description
`<cloud_peg_api_endpoint>`	Your API endpoint as described in PEG REST API on PrivaceraCloud[https://privacera.com/docs/en/peg-rest-api-on-privaceracloud.html]
`<api_key>`	Your API key, as described in API Key on PrivaceraCloud[https://privacera.com/docs/en/api-key-on-privaceracloud.html].
`schemelist`	List of <encryption_schemes>.
`<encryption_scheme>`	One or more <encryption_schemes> to encrypt or decrypt data in datalist. See Encryption schemes[https://privacera.com/docs/en/encryption-schemes.html].
`datalist`	List of data elements, one for each scheme in the `schemelist` parameter.
`<data>`	A data element to be encrypted with `/protect` or decrypted with `/unprotect`.
`maskSchemeList`	List of data elements for masking, with at least one for each masking scheme in `maskSchemeList` parameter or more data elements to be masked.
`<masking_scheme>`	One or more <masking_schemes> to mask the data in `maskDataList`. See Masking schemes[https://privacera.com/document/preview/354287#UUID-afc57374-425e-4c23-2ae7-32c3bc9fd243].
`maskDataList`	List of data elements for masking, with at least one for each masking scheme in `maskSchemeList` parameter or more data elements to be masked.
`<application_user>`	The application user or end-user that connects to a service, such as Snowflake, UDF, or ODBC application. By way of scheme policies, the permission to use Privacera Encryption of this user is verified.

Line	Description
`presentationSchemeList`	Not shown here, the `/unprotect` request can include a field to specify an optional presentation scheme. On `/unprotect`, the server uses the `presentation_scheme` to obfuscate the data even more for display to authorized users. `presentationSchemeList` on `/protect` is ignored.

About constructing the datalist for /protect

Suppose you want to encrypt two database fields tagged with Privacera metadata `PERSON_NAME` and `EMAIL`. The value of your API `datalist` to encrypt can be constructed like this:

1. Extract from the database the unencrypted values of the tagged fields.
2. Format a JSON array of those values.
3. Call the API `/protect` endpoint to encrypt the values in that array.
4. Reformat the returned JSON array of the encrypted values to update the fields in your database.

About deconstructing the response from /unprotect

Suppose you want to decrypt two database fields tagged with Privacera metadata `PERSON_NAME` and `EMAIL`. The value of your API `datalist` to decrypt can be constructed like this:

1. Extract from the database the encrypted values of the tagged fields.
2. Format a JSON array of those values.
3. Call the API `/unprotect` endpoint to decrypt the values in that array.
4. Reformat the returned JSON array of the decrypted values to update the fields in your database.

Example of data transformation with /unprotect and presentation scheme

This example shows some original input data, its representation when encrypted, and its obfuscated result after decryption with `/unprotect` and an optional presentation scheme.

- Original value: `sally@gmail.com`
- Encrypted value: `xy12zb@1mno2.rtz`
- Value after decryption and presentation scheme. The domain portion has been obfuscated: `sally@ymxof.1dg`

Example PEG REST API endpoints for PrivaceraCloud

These examples do not show the `curl` or authentication or the PrivaceraCloud PEG API endpoint. Only the JSON bodies for the requests (with the curl `--data-raw` option) and responses are shown.

If you are testing with a self-signed certificate, to bypass the certificate validation check, add the `curl -k` option.

/protect with encryption scheme

The two elements in the input `datalist` array are encrypted with the encryption schemes[https://privacera.com/docs/en/encryption-schemes.html] `PERSON_NAME` and `EMAIL`.

```
--data-raw '{"schemelist":["PERSON_NAME",
"EMAIL"],
"datalist": [["Mark",
            "Jonathan","Christopher"],
           ["mark@example.com",
           "jonathan@test.com",
           "christopher@google.com"]],
           "user":"jimmybob@BigCo.com"}'
```

Response

```
"datalist":[["WjM5",
"5vpJF9zT",
"1EbplEYVBjy"],
["i0bD@WKbMYpr.CvE",
"?9aqS8zV@YUym.hkd",
"d501shhJEO&@YpvfOc.VYH"]],
"data":"",
"responseStatus:"SUCCESS"}
```

/protect with masking scheme

The element in the input `maskDataList` array is masked by the masking scheme `MASKING_SCHEME` .

This example uses the authentication token retrieved with `/authenticate` .

```
--data-raw '{
    "maskSchemelist": [
        "MASKING_SCHEME"
    ],
    "maskDatalist": [
       [
         "",null,"12-12-2012","12/12/2025T09:01:02"
       ]
           "user":"<application_user>"}'
```

Response

```
 {
    "maskDatalist": [
       [
      "",
            null,
            "**-**-****",
            "**/**/*******:**:**"
       ]
}
```

/protect with both encryption and masking

The element in the input `datalist` array is encrypted with the encryption scheme[https://privacera.com/docs/en/encryption-schemes.html] `SYSTEM_EMAIL` and at the same time the data in the input `maskDataList` is masked with the masking scheme[https://privacera.com/document/preview/354287#UUID-afc57374-425e-4c23-2ae7-32c3bc9fd243] `MASKING_SCHEME` .

```
--data-raw '{"schemelist":["SYSTEM_EMAIL"], \
"datalist":[
        ["sally@gmail.com"]
        ], \
      "maskSchemelist":["DATE_MASKING_SCHEME"], \
      "maskDatalist":[
        ["",null,"12-12-2012","12/12/2025T09:01:02"]
        ], \
"user":"padmin" }'
```

Response


```
{
    "datalist": [
        [
            "mNM-^@RUWqb.qRK"
        ]
    ],
    "data": "",
    "maskDatalist": [
        [
            "",
            null,
            "**-**-****",
            "**/**/*******:**:**"
        ]
    ],
    "responseStatus": "SUCCESS"
}
```


/unprotect without presentation scheme

The two elements in the input `datalist` array are decrypted with the encryption schemes[https://privacera.com/docs/en/encryption-schemes.html] `PERSON_NAME` and `EMAIL` .

```
--data-raw '{"schemelist":["PERSON_NAME", "EMAIL"],
"datalist":
[["WjM5","5vpJF9zT",
"1EbplEYVBjy"],
["i0bD@WKbMYpr.CvE",
"?9aqS8zV@YUym.hkd",
"d501shhJEO&@YpvfOc.VYH"]],
"user":"<application_user>"}'
```

Response

```
{"datalist":
[["Mark",
"Jonathan","Christopher"],
["mark@example.com",
"jonathan@test.com",
"christopher@google.com"]],
"data":"",
"responseStatus":"SUCCESS"}
```

/unprotect with Presentation Scheme

The input in the `datalist` array is decrypted with the encryption scheme[https://privacera.com/docs/en/encryption-schemes.html] `EMAIL2` and then obfuscated with the presentation scheme[https://privacera.com/docs/en/presentation-schemes.html] `EMAIL2_P` .

This example uses the authentication token retrieved with `/authenticate` .

```
--data-raw '{"datalist":[["8283a@QhbpH.yOs","5fGP@RyZBO.UZE"]],
           "schemelist":["EMAIL2"],
           "presentationSchemelist":["EMAIL2_P"]
           "user":"jimmybob@BigCo.com"}'
```

/unprotect with masking scheme

Masking schemes must not be used with `/unprotect` , which returns an error because the masked data cannot be unmasked.

Audit details for PEG REST API accesses

PrivaceraCloud records access to the PEG REST API encryption keys and schemes. For details, see Audits[https://privacera.com/docs/en/audits.html].

Was this helpful?

Yes No