Request Parameters: -

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter Name | M/O | Data Type | Description |
| data | M | String | JSON Payload needs to be encrypted with a random AES 256-bit symmetric key and pass it in the data parameter or tag. |
| key | M | String | The random AES-256-bit symmetric key need to encrypt with the public certificate, followed by the RSA-OAEP-256 algorithm, and pass it in the key parameter or tag. |
| bit | M | integer | 0(zero) – it’s a static value for all clients / 1- for millennial |

Example Request: -

{

"data": " JSON Payload needs to be encrypted with a random AES 256-bit symmetric key -- (String)”,

"key": " The random AES-256-bit symmetric key needs to encrypt with the public certificate, followed by

the RSA-OAEP-256 algorithm — (string)",

"bit": 0

}

Example AES-256-bit Symmetric key: - (Please don’t use this key’s this is only for your reference)

1. 41ed35d0934f0c3d0a9f3f36c5b89be60f7714f516381d43242f59299cd6e949
2. 5777441745751c16c46e6b56a44109df13c290d3c533b94c1b8a69d26d7d81a1

Sample encryption Request: -

{

    "data": "1D55AE30D705357474E785A4BBD843E62A777A3ED48C8426B34C93320C6DA9CB7EEC750BD88F67BA86FAD52CDDC57F59409BE05C8395A77EF00F1CCC1599EE92834E350205A66B05E950C49B68ADAEF443CAEE8387F720FF045D0B127115514445C165CBADE89696E0B9518088797D51C640149BFDB4DA56B4460849681E3E26FD1DA6C26A62455892F2A5C674249A26058BCECB5D9FC1C84DCD79967700513FC9A7DF2E021D100219CC3F11A076252DE08E6A8007C67ED973FD38053E333330D6CA54D0D1B79CE32D9C762231DEADEE8644A2E320B2E46F5E4451C3960202B4",

    "key": "",

    "bit": 0

}

Response Parameters: -

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter Name | M/O | Data Type | Description |
| data |  | String | API will send an encrypted response by using the same AES-256-bit symmetric key. which is used to encrypt the request JSON payload. |

NOTE: - Only in success scenarios & business failures API response will be in an encrypted format, for technical failures response of API is in JSON.

Example Response: -

{

"data": " API will send an encrypted response by using the same AES-256-bit symmetric key. which is used to encrypt the request JSON payload"

}

Sample Encrypted Response: -

{

    "data": "1D55AE30D705357474E785A4BBD843E62A777A3ED48C8426B34C93320C6DA9CB7EEC750BD88F67BA86FAD52CDDC57F59409BE05C8395A77EF00F1CCC1599EE92834E350205A66B05E950C49B68ADAEF443CAEE8387F720FF045D0B127115514445C165CBADE89696E0B9518088797D51C640149BFDB4DA56B4460849681E3E26FD1DA6C26A62455892F2A5C674249A26058BCECB5D9FC1C84DCD79967700513FC9A7DF2E021D100219CC3F11A076252DE08E6A8007C67ED973FD38053E333330D6CA54D0D1B79CE32D9C762231DEADEE8644A2E320B2E46F5E4451C3960202B4"

}

Please refer to the attached jar for encryption and decryption.



Decryption logic at our end: - (Reference)

var jose = require('jose');

var privateKey = 'indus-privatekey'; //referring private key

if (privateKey != null) {

context.request.body.readAsJSON(function (error, json) {

if (error) {

context.reject('CustomError', 'Request not in JSON');

} else {

var data = json.data

var bit = json.bit

var jweobj = jose.parse(json.key);

jweobj.setKey(privateKey);

jose.createJWEDecrypter(jweobj).decrypt(function (error, symmetricKey) { //decrypt the symmetric key using private key

if (error) {

context.reject('CustomError', 'JWE Decryption Failed with Certificate');

} else {

var aesHexKey

if (bit == 0){aesHexKey = symmetricKey}

if (bit == 1){aesHexKey = symmetricKey.toString('hex');}

context.set("mySymmetricKey", aesHexKey);

var jweobj = jose.parse(data);

jweobj.setKey('hex:' + aesHexKey);

jose.createJWEDecrypter(jweobj).decrypt(function (error, plaintext) { //decrypt the payload

if (error) {

context.reject('CustomError', 'JWE Decryption Failed with AES Key');

} else {

var decryptedJson = json

decryptedJson = JSON.parse(plaintext.toString());

context.message.body.write(decryptedJson);

}

});

}

});

}

});

}

Encryption Logic at our end: -

var jose = require('jose');

var jweHdr = jose.createJWEHeader('A256GCM'); //supported algorithm for encryption

var mySymmetricKey = context.get('mySymmetricKey');

jweHdr.setProtected('alg', 'A256KW');

jweHdr.setKey('hex:' + mySymmetricKey);

context.message.body.readAsJSON(function (error, json) {

if (error) {

context.reject('CustomError', 'Response not in JSON');

} else{

var \_my\_json\_string = JSON.stringify(json);

let \_my\_json\_buffer = new Buffer(\_my\_json\_string);

jose.createJWEEncrypter(jweHdr).update(\_my\_json\_buffer).encrypt('compact', function (error, jweObj) {

if (error) {

context.reject('CustomError', 'JWE Encryption Failed');

} else {

context.message.body.write({"data": jweObj});

}

}

);

}

}

);

IBM Knowledge Centre reference: -

https://www.ibm.com/docs/en/datapower-gateway/10.0.1?topic=apis-jose-module