Banco Azteca

Integración de cifrado en Features

GOBIERNO DE API'S

Contenido

Intregración de cambios en Integration Test

Integración de cambios en Api Proxy

Intregración de cambios en Integration Test

Api de Seguridad

La siguiente documentación se basa en cifrado de información de los recursos de las Apis que estemos desarrollando teniendo como base el Api de Seguridad a nivel área.

Esta Api de seguridad es la encargada de proveer el par de llaves asimétricas que nos permitirán cifrar y descifrar información sensible para los recursos de nuestras Apis.

Scenario en Feature para obtención de llaves asimétricas

Antes de probar los distintos Scenarios de los recursos de nuestras Apis, en los features debemos agregar un éste Scenario el cuál consumirá el recurso **GET /aplicaciones/llaves** del Api de Seguridad para obtener la siguiente información e interactuar con datos sensibles:

```
Scenario: Genera token y llaves asimétricas
        Given I set bearer token
        And I have valid client TLS configuration
        And I set x-ismock header to true
        When I GET `apigeeDomain`/capital-humano/desarrollo-personal-informacion/seguridad/v1/aplicaciones/llaves
            Then response code should be 200
            And response body should be valid json
            And I store the value of body path $.resultado.idAcceso as idAccess in global scope
            And I store the value of body path $.resultado.accesoPrivado as privateKey in global scope
             And I store the value of body path $.resultado.accesoPublico as publicKey in global scope
            And I store the value of body path $.resultado.integrationTest as localTest in global scope
```

1. **idAccess**: Identificador del acceso generado en backend para la interacción con los 2 pares de llaves asimétricas generadas para realizar las operaciones de cifrado y descifrado tanto del lado del cliente como del backend.
2. **privateKey**: Llave asimétrica privada para descifrar la información que regrese el backend.
3. **publicKey**: Llave asimétrica pública para cifrar la información que enviaremos al backend.
4. **localTest**: Bandera que indica si las pruebas con features cifrados se están realizando en el proyecto de IntegrationTest o en el Api Proxy.
    a. **Cuando la bandera está en true sobre el proyecto de IntegrationTest no cifra ni descifra información sensible**
    b. **Cuando repliquemos los cambios al Api Proxy la bandera tomará el valor a null/false por lo que si ejecutará las operaciones de cifrado y descifrado.**

Mock de respuesta exitosa para obtención de llaves

Cuando trabajemos en IntegrationTest el mock que responde al anterior escenario tiene la siguiente forma:

llaves_ok.json


```
{
  "mensaje": "Operación Exitosa",
  "folio": "17920190617173411592",
  "resultado": {
      "idAcceso": "265",
      "accesoPublico": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCXdmNZuPxnMs31M7nsnmfyKihcqU4WMg/SFQq5IAQ7aA+GI06IG/
5268jWDZVoO2INM/X8/rVhHi+22XzG28k1EqLXlsg588Ln2O1OylxugSO7fQpJz32huKqlN4FgRMTGIHqMPgpsKmdvNUwtE3j5qrv5PTK5J/pvOEmE
MmTM9QIDAQAB",
      "accesoPrivado": "MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBAI9/o9Ct9TpgBeiGr7xh00LgGrTeAWmcQOPZY9vrbX
Iq+Pkex9mQpf7zphLRI3HZbIwvwpGPKQAD54Q+rISse8KOtMmA2qDweIj8Qh+ZtY9QOtvFAJYN15LxMVF6t3/SKcE/tk6mcMzakYY01wZp/JbcR4N8
idM5Bo3yNQzyNMa5AgMBAAECgYAsytHurK5Uh26ws4IdkoeoY13SwU29kpmqxcvm9OkX7xQCZzpIzl8SYymezAt9WToJ1Equtvo8V/uniwy86T5D9b
zMmFXmsn0ktLea2N9x0mVSrGVH+I+qba92hagWiOZiW/zsnovZLdazdYPIg43CsI1DHrE7q7fvGV4/WXgLgQJBAOqjIAQg2MODlHjbVEmJ9n/3BmR1
4 ==",
      "integrationTest": true
  }
}
```


Visualización de recurso de llaves en la configuración de Apimocker

En el config-generated se ve de la siguiente manera el recurso de **GET /aplicaciones/llaves**:

```
"basepath_del_area_dónde_quedará_el_api_de_seguridad/seguridad/v1/aplicaciones/llaves":{
            "latency": 10,
            "verbs":[
                "get"
            ],
            "switch": [
            ],
            "responses": {
                "get": {
                    "httpStatus": 200,
                    "mockFile": "llaves_ok.json"
                }
            }
        }
```

1. Caben mencionar que el mapeo de éste recurso de llaves en el config-generated así como el mock de éste recurso **no se aplica** al replicar los cambios al Api Proxy conservando sólo el Scenario en los features.

Una vez que obtenemos las llaves para interactuar con el cifrado y descifrado de información dentro del proyecto de IntegrationTest, tenemos que agregar nuevas funciones de Gherkin para realizar dichas operaciones:

La siguiente información muestra ejemplos para la utilización de funciones de Gherkin para cifrar y descifrar información sensible, cabe mencionar que los valores en los payloads tanto de entrada(request) como de salida(response) deben ir en claro, así como se muestran en las respectivas especificaciones de nuestras Apis.

Descifrado de Información (response)

Ejemplo: GET/*/beneficiarios: (**descifrado**)

La función de éste recurso de ejemplo es obtener los beneficiarios que dio de alta un empleado para un seguro de vida, por lo que la información de la respuesta cuenta con datos sensibles que se tienen que descifrar para validar su valor con regex que colocamos en nuestros features.

Para descifrar la información en la respuesta JSON se debe obtener el **id de acceso** y la **llave privada** del recurso de **GET /aplicaciones/llaves** del **Api de Seguridad** como se vio anteriormente.

En nuestro Scenario para obtener los beneficiarios de un empleado tenemos que agregar lo siguiente.

1. Setear en el header **x-idAcceso** el idAccess que recuperamos anteriormente de la siguiente forma:

```
And I set x-idAcceso header to `idAccess`
```

2. Para descifrar la información de respuesta, en éste caso el nombre del beneficiario tenemos que hacerlo con la siguiente función Gherkin al momento de validar la respuesta dentro de nuestro Scenario

```
And deciphering with the key privateKey the response field $.resultado.beneficiarios.principales[*].nombre and loc
alTest as environment and RSA_PKCS1_PADDING as encryption algorithm should be ^[A-Za-záéíóúÁÉÍÓÚÑñÜü0-9.,-
\s:]{1,255}$
```

   a. Se envía la llave privada.
   b. Se indica con jsonPath el campo a descifrar.
   c. Se envía la bandera que indica si las pruebas se hacen en el proyecto de IntegrationTest.
   d. Se envía el algoritmo de encripción.
   e. Al final se coloca la expresión regular con la cuál va a validar el valor obtenido del descifrado.

Nuestro Scenario completo queda de la siguiente manera:

```
Scenario Outline: Obtener los beneficiarios de seguro de vida de un empleado
        Given I set bearer token
        And I set x-idAcceso header to `idAccess`
        And I have valid client TLS configuration

        When I GET `apigeeDomain`/capital-humano/desarrollo-personal-
informacion/empleados/`deploymentSuffix`/<numeroEmpleado>/beneficiarios
            Then response code should be 200
            And response body should be valid json
            And response body path $.mensaje should be Operación Exitosa
            And response body path $.folio should be ^\d{6,20}$
            And response body path $.resultado.beneficiarios.principales[*].idConsecutivo should be ^\d{1,5}$
            And deciphering with the key privateKey the response field $.resultado.beneficiarios.principales[*].no
mbre and localTest as environment and RSA_PKCS1_PADDING as encryption algorithm should be ^[A-Za-záéíóúÁÉÍÓÚÑñÜü0-
9.,-\s:]{1,255}$
            And deciphering with the key privateKey the response field $.resultado.beneficiarios.principales[*].ap
ellidoPaterno and localTest as environment and RSA_PKCS1_PADDING as encryption algorithm should be ^[A-Za-
záéíóúÁÉÍÓÚÑñÜü0-9.,-\s:]{1,255}$
```

```
            And deciphering with the key privateKey the response field $.resultado.beneficiarios.principales[*].ap
ellidoMaterno and localTest as environment and RSA_PKCS1_PADDING as encryption algorithm should be ^[A-Za-
záéíóúÁÉÍÓÚÑñÜü0-9.,-\s:]{1,255}$
            And response body path $.resultado.beneficiarios.principales[*].fechaNacimiento should be ^[0-
9]{2}[/][0-9]{2}[/][0-9]{4}
            And response body path $.resultado.beneficiarios.principales[*].genero should be ^[FM]
            And response body path $.resultado.beneficiarios.principales[*].porcentaje should be ^\d{1,3}$
            And response body path $.resultado.beneficiarios.principales[*].idParentesco should be ^\d{1,2}$
            And response body path $.resultado.beneficiarios.principales[*].parentesco should be ^[A-Za-
zñáéíóúÁÉÍÓÚÑ0-9.(),-\s:]{0,255}$
        Examples:
            | numeroEmpleado |
            | 211027         |
```

Cifrado de Información (Request)

Ejemplo: PUT/*/beneficiarios: (**cifrado**)

La función de éste recurso es dar de alta beneficiarios de seguro de vida de un empleado, por lo que cierta información de entrada es sensible y tiene que ir cifrada.

Para cifrar la información del request JSON se debe obtener el **id de acceso** y la **llave pública** del recurso de **GET /aplicaciones/llaves** del **Api de Seguridad** como se vio anteriormente.

En nuestro Scenario para dar de alta los beneficiarios de un empleado tenemos que agregar lo siguiente.

1. Setear en el header **x-idAcceso** el idAccess que recuperamos anteriormente de la siguiente forma:

```
And I set x-idAcceso header to `idAccess`
```

2. Para identificar los campos a cifrar: “nombre”, “apellidoPaterno”, “apellidoMaterno”, se utilizan la función:

```
And I need to encrypt the parameters {beneficiarios.principales[*].nombre,beneficiarios.principales[*].apellidoPat
erno,beneficiarios.principales[*].apellidoMaterno}
```

3. La función Gherkin para cifrar los valores de los campos sensibles es la siguiente:

```
And I use the encryption algorithm RSA_PKCS1_PADDING and localTest as environment and the key publicKey for prepar
e a body as {"beneficiarios":{"principales":[{"idConsecutivo":<principalConsec>, "nombre":<principalNombre>,"apell
idoPaterno":<principalPaterno>,"apellidoMaterno":<principalMaterno>,"fechaNacimiento":<principalFecha>,"genero":<p
rincipalGenero>,"idParentesco":<principalPatentesco>,"porcentaje":<principalPorcentaje>}]}}
```

   a. Se envía el algoritmo de encripción.
   b. Se envía la bandera que indica si las pruebas se hacen en el proyecto de IntegrationTest.
   c. Se envía la llave pública.
   d. Se envía el body con valores en claro, con la estructura que se muestra en la especificación del Api.

Como bien saben las variables con operador diamante toman valores en tiempo de ejecución de los examples de cada Scenario.

Nuestro Scenario completo queda de la siguiente manera:

```
Scenario Outline: Verificar que responda un mensaje exitoso al actualizar los beneficiarios
        Given I set Content-Type header to application/json
        And I set bearer token
        And I set x-idAcceso header to `idAccess`
        And I need to encrypt the parameters {beneficiarios.principales[*].nombre,beneficiarios.principales[*].ape
llidoPaterno,beneficiarios.principales[*].apellidoMaterno}
        And I use the encryption algorithm RSA_PKCS1_PADDING and localTest as environment and the key publicKey fo
r prepare a body as {"beneficiarios":{"principales":[{"idConsecutivo":<principalConsec>, "nombre":<principalNombre
>,"apellidoPaterno":<principalPaterno>,"apellidoMaterno":<principalMaterno>,"fechaNacimiento":<principalFecha>,"ge
nero":<principalGenero>,"idParentesco":<principalPatentesco>,"porcentaje":<principalPorcentaje>}]}}
        And I have valid client TLS configuration

        When I PUT `apigeeDomain`/capital-humano/desarrollo-personal-
informacion/empleados/`deploymentSuffix`/<numeroEmpleado>/beneficiarios
            Then response code should be 200
            And response body should be valid json
            And response body path $.mensaje should be Operación Exitosa
            And response body path $.folio should be ^\d{6,20}$
            And response body path $.resultado.folioSolicitud should be ^\d{1,20}$
        Examples:
        | numeroEmpleado | principalConsec | principalNombre  | principalPaterno | principalMaterno | principalFec
ha | principalGenero | principalPatentesco | principalPorcentaje |
        | 211027         | 0               | "JULIETA MARTHA" | "GARCIA"         | "GARCIA"         | "15/06/1968"
   | "F"             | 12                  | 100                 |
```

En resumen se muestra a continuación como quedan los features completos ejemplificando el cifrado y descifrado de información:

Ejemplo de Feature completo para descifrado de Información

```
Feature: Consultar los beneficiarios de seguro de vida registrados en SAP del empleado
    Consulta los beneficiarios del empleado

    Scenario: Negocio un consumer key y secret key de la app de prueba
        Given I have basic authentication credentials `apigeeUsername` and `apigeePassword`
        And I have valid client TLS configuration

        When I GET `apigeeHost`/v1/organizations/`apigeeOrg`/developers/`apigeeDeveloper`/apps/`apigeeApp`
            Then response code should be 200
            And response body should be valid json
            And I store the value of body path $.credentials[0].consumerKey as globalConsumerKey in global scope
            And I store the value of body path $.credentials[0].consumerSecret as globalConsumerSecret in global s
cope

    Scenario: Negocia un access token con el Authorization server
        Given I set form parameters to
        | parameter | value |
        | grant_type | client_credentials |
        And I have basic authentication credentials `globalConsumerKey` and `globalConsumerSecret`
        And I have valid client TLS configuration

        When I POST to `apigeeDomain`/`apigeeOauthEndpoint`
            Then response code should be 200
            And response body should be valid json
            And I store the value of body path $.access_token as access token

    Scenario: Genera token y llaves asimétricas
        Given I set bearer token
        And I have valid client TLS configuration
        And I set x-ismock header to true
        When I GET `apigeeDomain`/capital-humano/desarrollo-personal-informacion/seguridad/v1/llaves
```

```
            Then response code should be 200
            And response body should be valid json
            And I store the value of body path $.resultado.idAcceso as idAccess in global scope
            And I store the value of body path $.resultado.accesoPrivado as privateKey in global scope
            And I store the value of body path $.resultado.integrationTest as localTest in global scope

    Scenario Outline: Obtener los beneficiarios de seguro de vida de un empleado
        Given I set bearer token
        And I set x-idAcceso header to `idAccess`
        And I have valid client TLS configuration

        When I GET `apigeeDomain`/capital-humano/desarrollo-personal-
informacion/empleados/`deploymentSuffix`/<numeroEmpleado>/beneficiarios
            Then response code should be 200
            And response body should be valid json
            And response body path $.mensaje should be Operación Exitosa
            And response body path $.folio should be ^\d{6,20}$
            And response body path $.resultado.beneficiarios.principales[*].idConsecutivo should be ^\d{1,5}$
            And deciphering with the key privateKey the response field $.resultado.beneficiarios.principales[*].no
mbre and localTest as environment and RSA_PKCS1_PADDING as encryption algorithm should be ^[A-Za-záéíóúÁÉÍÓÚÑñÜü0-
9.,-\s:]{1,255}$
            And deciphering with the key privateKey the response field $.resultado.beneficiarios.principales[*].ap
ellidoPaterno and localTest as environment and RSA_PKCS1_PADDING as encryption algorithm should be ^[A-Za-
záéíóúÁÉÍÓÚÑñÜü0-9.,-\s:]{1,255}$
            And deciphering with the key privateKey the response field $.resultado.beneficiarios.principales[*].ap
ellidoMaterno and localTest as environment and RSA_PKCS1_PADDING as encryption algorithm should be ^[A-Za-
záéíóúÁÉÍÓÚÑñÜü0-9.,-\s:]{1,255}$
            And response body path $.resultado.beneficiarios.principales[*].fechaNacimiento should be ^[0-
9]{2}[/][0-9]{2}[/][0-9]{4}
            And response body path $.resultado.beneficiarios.principales[*].genero should be ^[FM]
            And response body path $.resultado.beneficiarios.principales[*].porcentaje should be ^\d{1,3}$
            And response body path $.resultado.beneficiarios.principales[*].idParentesco should be ^\d{1,2}$
            And response body path $.resultado.beneficiarios.principales[*].parentesco should be ^[A-Za-
zñáéíóúÁÉÍÓÚÑ0-9.(),-\s:]{0,255}$
        Examples:
            | numeroEmpleado |
            | 211027         |

    Scenario Outline: Verificar que responda un mensaje de error cuando se envie una solicitud con parámetros invá
lidos
        Given I set bearer token
        And I set x-idAcceso header to `idAccess`
        And I have valid client TLS configuration

        When I GET `apigeeDomain`/capital-humano/desarrollo-personal-
informacion/empleados/`deploymentSuffix`/<numeroEmpleado>/beneficiarios
            Then response code should be 400
            And response body should be valid json
            And response body path $.codigo should be ^400\.CH-Desarollo-Personal-Informacion-Empleados\.\d{4}$
            And response body path $.mensaje should be ^[A-Za-záéíóúÁÉÍÓÚ0-9.,-\s:]{1,255}$
            And response body path $.folio should be ^\d{6,20}$
            And response body path $.info should be ^https:\/\/baz-
developer\.bancoazteca\.com\.mx\/\w{4,6}#400.CH-Desarollo-Personal-Informacion-Empleados.\d{4}$
            And response body path $.detalles[0] should be ^[A-Za-záéíóúÁÉÍÓÚ0-9.,-\s:]{1,255}$
        Examples:
            | numeroEmpleado |
            | null           |
            | notFound       |
            | @              |

    Scenario Outline: Verificar que responda un mensaje de error cuando el x-idAcceso ha expirado
        Given I set bearer token
        And I set x-idAcceso header to `idAccess`
        And I have valid client TLS configuration

        When I GET `apigeeDomain`/capital-humano/desarrollo-personal-
informacion/empleados/`deploymentSuffix`/<numeroEmpleado>/beneficiarios
            Then response code should be 400
```

```
            And response body should be valid json
            And response body path $.codigo should be ^400\.CH-Desarollo-Personal-Informacion-Empleados\.\d{4}$
            And response body path $.mensaje should be ^[A-Za-záéíóúÁÉÍÓÚ0-9.,-\s:]{1,255}$
            And response body path $.folio should be ^\d{6,20}$
            And response body path $.info should be ^https:\/\/baz-
developer\.bancoazteca\.com\.mx\/\w{4,6}#400.CH-Desarollo-Personal-Informacion-Empleados.\d{4}$
            And response body path $.detalles[0] should be ^[A-Za-záéíóúÁÉÍÓÚ0-9.,-\s:]{1,255}$
        Examples:
            | numeroEmpleado |
            | 64036          |

    Scenario Outline: Verificar que responda un mensaje de error cuando no se encuentre información 404
        Given I set bearer token
        And I set x-idAcceso header to `idAccess`
        And I have valid client TLS configuration

        When I GET `apigeeDomain`/capital-humano/desarrollo-personal-
informacion/empleados/`deploymentSuffix`/<numeroEmpleado>/beneficiarios
            Then response code should be 404
            And response body should be valid json
            And response body path $.codigo should be ^404\.CH-Desarollo-Personal-Informacion-Empleados\.\d{4}$
            And response body path $.mensaje should be ^[A-Za-záéíóúÁÉÍÓÚ0-9.,-\s:]{1,255}$
            And response body path $.folio should be ^\d{6,20}$
            And response body path $.info should be ^https:\/\/baz-
developer\.bancoazteca\.com\.mx\/\w{4,6}#404.CH-Desarollo-Personal-Informacion-Empleados.\d{4}$
            And response body path $.detalles[0] should be ^[A-Za-záéíóúÁÉÍÓÚ0-9.,-\s:]{1,255}$
        Examples:
            | numeroEmpleado |
            | 761577         |

    Scenario Outline: Verificar que responda un mensaje de error cuando ocurrió un problema interno en la aplicaci
ón 500
        Given I set bearer token
        And I set x-idAcceso header to `idAccess`
        And I set x-ismock header to true
        And I have valid client TLS configuration

        When I GET `apigeeDomain`/capital-humano/desarrollo-personal-
informacion/empleados/`deploymentSuffix`/<numeroEmpleado>/beneficiarios
            Then response code should be 500
            And response body should be valid json
            And response body path $.codigo should be ^500\.CH-Desarollo-Personal-Informacion-Empleados\.\d{4}$
            And response body path $.mensaje should be ^[A-Za-záéíóúÁÉÍÓÚ0-9.,-\s:]{1,255}$
            And response body path $.folio should be ^\d{6,20}$
            And response body path $.info should be ^https:\/\/baz-
developer\.bancoazteca\.com\.mx\/\w{4,6}#500.CH-Desarollo-Personal-Informacion-Empleados.\d{4}$
            And response body path $.detalles[0] should be ^[A-Za-záéíóúÁÉÍÓÚ0-9.,-\s:]{1,255}$
        Examples:
            | numeroEmpleado |
            | 176286         |
```

Ejemplo de Feature completo para cifrado de Información

```
Feature: Guardar o actualizar los beneficiarios de seguro de vida en SAP
    Guarda o actualiza los beneficiarios generando un numero de folio

    Scenario: Negocio un consumer key y secret key de la app de prueba
        Given I have basic authentication credentials `apigeeUsername` and `apigeePassword`
        And I have valid client TLS configuration

        When I GET `apigeeHost`/v1/organizations/`apigeeOrg`/developers/`apigeeDeveloper`/apps/`apigeeApp`
            Then response code should be 200
            And response body should be valid json
            And I store the value of body path $.credentials[0].consumerKey as globalConsumerKey in global scope
            And I store the value of body path $.credentials[0].consumerSecret as globalConsumerSecret in global s
cope

    Scenario: Negocia un access token con el Authorization server
        Given I set form parameters to
        | parameter | value |
        | grant_type | client_credentials |
        And I have basic authentication credentials `globalConsumerKey` and `globalConsumerSecret`
        And I have valid client TLS configuration

        When I POST to `apigeeDomain`/`apigeeOauthEndpoint`
            Then response code should be 200
            And response body should be valid json
            And I store the value of body path $.access_token as access token

    Scenario: Genera token y llaves asimétricas
        Given I set bearer token
        And I have valid client TLS configuration
        And I set x-ismock header to true
        When I GET `apigeeDomain`/capital-humano/desarrollo-personal-informacion/seguridad/v1/llaves
            Then response code should be 200
            And response body should be valid json
            And I store the value of body path $.resultado.idAcceso as idAccess in global scope
            And I store the value of body path $.resultado.accesoPublico as publicKey in global scope
            And I store the value of body path $.resultado.integrationTest as localTest in global scope

    Scenario Outline: Verificar que responda un mensaje exitoso al actualizar los beneficiarios
        Given I set Content-Type header to application/json
        And I set bearer token
        And I set x-idAcceso header to `idAccess`
        And I need to encrypt the parameters {beneficiarios.principales[*].nombre,beneficiarios.principales[*].ape
llidoPaterno,beneficiarios.principales[*].apellidoMaterno}
        And I use the encryption algorithm RSA_PKCS1_PADDING and localTest as environment and the key publicKey fo
r prepare a body as {"beneficiarios":{"principales":[{"idConsecutivo":<principalConsec>, "nombre":<principalNombre
>,"apellidoPaterno":<principalPaterno>,"apellidoMaterno":<principalMaterno>,"fechaNacimiento":<principalFecha>,"ge
nero":<principalGenero>,"idParentesco":<principalPatentesco>,"porcentaje":<principalPorcentaje>}]}}
        And I have valid client TLS configuration

        When I PUT `apigeeDomain`/capital-humano/desarrollo-personal-
informacion/empleados/`deploymentSuffix`/<numeroEmpleado>/beneficiarios
            Then response code should be 200
            And response body should be valid json
            And response body path $.mensaje should be Operación Exitosa
            And response body path $.folio should be ^\d{6,20}$
            And response body path $.resultado.folioSolicitud should be ^\d{1,20}$
        Examples:
        | numeroEmpleado | principalConsec | principalNombre  | principalPaterno | principalMaterno | principalFec
ha | principalGenero | principalPatentesco | principalPorcentaje |
        | 211027         | 0               | "JULIETA MARTHA" | "GARCIA"         | "GARCIA"         | "15/06/1968"
   | "F"             | 12                  | 100                  |

    Scenario Outline: Verificar que responda un mensaje de error por falta de parámetros requeridos
        Given I set Content-Type header to application/json
        And I set bearer token
```

```
        And I set x-idAcceso header to `idAccess`
        And I need to encrypt the parameters {beneficiarios.principales[*].nombre,beneficiarios.principales[*].ape
llidoPaterno,beneficiarios.principales[*].apellidoMaterno}
        And I use the encryption algorithm RSA_PKCS1_PADDING and localTest as environment and the key publicKey fo
r prepare a body as {"beneficiarios":{"principales":[{"idConsecutivo":<principalConsec>, "apellidoPaterno":<princi
palPaterno>,"apellidoMaterno":<principalMaterno>,"fechaNacimiento":<principalFecha>,"genero":<principalGenero>,"id
Parentesco":<principalPatentesco>,"porcentaje":<principalPorcentaje>}]}}
        And I have valid client TLS configuration

        When I PUT `apigeeDomain`/capital-humano/desarrollo-personal-
informacion/empleados/`deploymentSuffix`/<numeroEmpleado>/beneficiarios
            Then response code should be 400
            And response body should be valid json
            And response body path $.codigo should be ^400\.CH-Desarollo-Personal-Informacion-Empleados\.\d{4}$
            And response body path $.mensaje should be ^[A-Za-záéíóúÁÉÍÓÚ0-9.,-\s:]{1,255}$
            And response body path $.folio should be ^\d{6,20}$
            And response body path $.info should be ^https:\/\/baz-
developer\.bancoazteca\.com\.mx\/\w{4,6}#400.CH-Desarollo-Personal-Informacion-Empleados.\d{4}$
            And response body path $.detalles[0] should be ^[A-Za-záéíóúÁÉÍÓÚ0-9.,-\s:]{1,255}$
        Examples:
        | numeroEmpleado | principalConsec | principalPaterno | principalMaterno | principalFecha | principalGener
o | principalPatentesco | principalPorcentaje |
        | 211027         | 0               | "GARCIA"         | "GARCIA"         | "15/06/1968"   | "F"
  | 12                 | 100               |

    Scenario Outline: Verificar que responda un mensaje de error por parámetros inválidos
        Given I set Content-Type header to application/json
        And I set bearer token
        And I set x-idAcceso header to `idAccess`
        And I need to encrypt the parameters {beneficiarios.principales[*].nombre,beneficiarios.principales[*].ape
llidoPaterno,beneficiarios.principales[*].apellidoMaterno}
        And I use the encryption algorithm RSA_PKCS1_PADDING and localTest as environment and the key publicKey fo
r prepare a body as {"beneficiarios":{"principales":[{"idConsecutivo":<principalConsec>, "nombre":<principalNombre
>,"apellidoPaterno":<principalPaterno>,"apellidoMaterno":<principalMaterno>,"fechaNacimiento":<principalFecha>,"ge
nero":<principalGenero>,"idParentesco":<principalPatentesco>,"porcentaje":<principalPorcentaje>}]}}
        And I have valid client TLS configuration

        When I PUT `apigeeDomain`/capital-humano/desarrollo-personal-
informacion/empleados/`deploymentSuffix`/<numeroEmpleado>/beneficiarios
            Then response code should be 400
            And response body should be valid json
            And response body path $.codigo should be ^400\.CH-Desarollo-Personal-Informacion-Empleados\.\d{4}$
            And response body path $.mensaje should be ^[A-Za-záéíóúÁÉÍÓÚ0-9.,-\s:]{1,255}$
            And response body path $.folio should be ^\d{6,20}$
            And response body path $.info should be ^https:\/\/baz-
developer\.bancoazteca\.com\.mx\/\w{4,6}#400.CH-Desarollo-Personal-Informacion-Empleados.\d{4}$
            And response body path $.detalles[0] should be ^[A-Za-záéíóúÁÉÍÓÚ0-9.,-\s:]{1,255}$
        Examples:
        | numeroEmpleado | principalConsec | principalNombre  | principalPaterno | principalMaterno | principalFec
ha | principalGenero | principalPatentesco | principalPorcentaje |
        | 211027         | 0               | "JULIETA MARTHA" | "GARCIA"         | "GARCIA"         | "15/06/1968"
   | "MUJER"          | "MADRE"             | "CIEN"               |

    Scenario Outline: Verificar que responda un mensaje de error cuando ocurrió un problema interno en la aplicaci
ón 500
        Given I set Content-Type header to application/json
        And I set bearer token
        And I set x-idAcceso header to `idAccess`
        And I set x-ismock header to true
        And I need to encrypt the parameters {beneficiarios.principales[*].nombre,beneficiarios.principales[*].ape
llidoPaterno,beneficiarios.principales[*].apellidoMaterno}
        And I use the encryption algorithm RSA_PKCS1_PADDING and localTest as environment and the key publicKey fo
r prepare a body as {"beneficiarios":{"principales":[{"idConsecutivo":<principalConsec>, "nombre":<principalNombre
>,"apellidoPaterno":<principalPaterno>,"apellidoMaterno":<principalMaterno>,"fechaNacimiento":<principalFecha>,"ge
nero":<principalGenero>,"idParentesco":<principalPatentesco>,"porcentaje":<principalPorcentaje>}]}}
        And I have valid client TLS configuration
```

```
        When I PUT `apigeeDomain`/capital-humano/desarrollo-personal-
informacion/empleados/`deploymentSuffix`/<numeroEmpleado>/beneficiarios
            Then response code should be 500
            And response body should be valid json
            And response body path $.codigo should be ^500\.CH-Desarollo-Personal-Informacion-Empleados\.\d{4}$
            And response body path $.mensaje should be ^[A-Za-záéíóúÁÉÍÓÚ0-9.,-\s:]{1,255}$
            And response body path $.folio should be ^\d{6,20}$
            And response body path $.info should be ^https:\/\/baz-
developer\.bancoazteca\.com\.mx\/\w{4,6}#500.CH-Desarollo-Personal-Informacion-Empleados.\d{4}$
            And response body path $.detalles[0] should be ^[A-Za-záéíóúÁÉÍÓÚ0-9.,-\s:]{1,255}$
        Examples:
        | numeroEmpleado | principalConsec | principalNombre  | principalPaterno | principalMaterno | principalFec
ha | principalGenero | principalPatentesco | principalPorcentaje |
        | "xqxq"         | 0               | "JULIETA MARTHA" | "GARCIA"         | "GARCIA"         | "15/06/1968"
   | "F"             | 12                  | 100                 |
```

Integración de cambios en Api Proxy

Consideraciones generales

Como bien sabemos se replica lo siguiente al Api Proxy:

1. Mocks
2. Configuración del Apimocker (recursos sin basePath)
3. Features
    a. En el Scenario que recupera las llaves asimétricas aquí no es necesaria la siguiente instrucción:

```
And I store the value of body path $.resultado.integrationTest as localTest in global scope
```

Además de eso necesitamos incluir lo siguiente para realizar una integración completa de los features con cifrado y descifrado.

Cambios en Api de Seguridad

En el Api de Seguridad

1. Crear política **FC-Generacion-Llaves** con el siguiente contenido:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<FlowCallout async="false" continueOnError="false" enabled="true" name="FC-
Generacion-Llaves">
  <DisplayName>FC-Generacion-Llaves</DisplayName>
  <FaultRules/>
  <Properties/>
  <SharedFlowBundle>sf-generacion-llaves</SharedFlowBundle>
</FlowCallout>
```

2. Modificar el TargetEndpoint **mock.xml** para incluir la llamada al sharedflow **sf-generacion-llaves** a través de la politica llamada **FC-Generacion-Llaves** dentro del Flow de nuestro recurso de **GET /aplicaciones/llaves** como se muestra a continuación:

```
<Flows>
    <Flow name="GET /aplicaciones/llaves">
        <Condition>(proxy.pathsuffix MatchesPath "/aplicaciones/llaves")
and (request.verb = "GET")</Condition>
        <Description/>
        <Request/>
        <Response>
            <Step>
                <Name>FC-Generacion-Llaves</Name>
            </Step>
        </Response>
    </Flow>
</Flows>
```

Ésta configuración hace lo siguiente:

Al llamar al recurso **GET /aplicaciones/llaves** se llama al Shared Flow **sf-generacion-llaves** el cuál internamente crea 2 pares de llaves asimétricas jwt, un id de acceso y un tiempo de vida de éste último (1 hr), ésta información se guarda en cache de Apigee para después recuperarla para el cifrado y descifrado de información.

Cambios en Api a integrar (rama)

1. Crear política **FC-Cifrado-Acceso** con el siguiente contenido:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<FlowCallout async="false" continueOnError="false" enabled="true" name="FC-Cifrado-Acceso">
  <DisplayName>FC-Cifrado-Acceso</DisplayName>
  <FaultRules/>
  <Properties/>
  <SharedFlowBundle>sf-cifrado-acceso</SharedFlowBundle>
</FlowCallout>
```

2. Modificar el TargetEndpoint **mock.xml** para incluir la llamada al sharedflow **sf-cifrado-acceso** a través de la politica llamada **FC-Cifrado-Acceso**, dentro del PreFlow-Request y el PostFlow-Response como se muestra a continuación:

   a. **PreFlow**

```
<PreFlow name="PreFlow">
    <Request>
      <!-- Begin Políticas -->
      <Step>
        <Name>FC-Cifrado-Acceso</Name>
      </Step>
      <!-- End Políticas -->
      <!-- Begin Logging -->
      <Step>
        <Name>JS-ContextFlow</Name>
      </Step>
```

```
        <Step>
          <Name>FC-Logging</Name>
        </Step>
        <!-- End Logging -->
      </Request>
      <Response>
        <!-- Begin Logging -->
        <Step>
          <Name>JS-ContextFlow</Name>
        </Step>
        <Step>
          <Name>FC-Logging</Name>
        </Step>
        <!-- End Logging -->
        <!-- Begin Políticas -->
        <!-- End Políticas -->
      </Response>
    </PreFlow>
```

b. PostFlow

```
<PostFlow name="PostFlow">
    <Request>
      <!-- Begin Políticas -->
      <!-- End Políticas -->
      <!-- Begin Logging -->
      <Step>
        <Name>JS-ContextFlow</Name>
      </Step>
      <Step>
        <Name>FC-Logging</Name>
      </Step>
      <!-- End Logging -->
    </Request>
    <Response>
      <!-- Begin Conext Flow -->
      <Step>
        <Name>JS-ContextFlow</Name>
      </Step>
      <!-- End Conext Flow -->
      <!-- Begin Políticas -->
      <Step>
        <Name>FC-Cifrado-Acceso</Name>
      </Step>
      <Step>
        <Condition>(message.status.code GreaterThanOrEquals
400)</Condition>
        <Name>AM-Error-Mock</Name>
      </Step>
      <Step>
```

```
            <Condition>(message.status.code GreaterThanOrEquals
400)</Condition>
            <Name>RF-Error-Mock</Name>
          </Step>
          <Step>
            <Name>AM-Eliminar-Headers-Mock</Name>
          </Step>
          <!-- End Políticas -->
        </Response>
      </PostFlow>
```

Ésta configuración hace lo siguiente:

1. El llamado del Shared Flow **sf-cifrado-acceso** en el **PreFlow-Request** descifra la información que viene de la ejecución del feature el cuál cifró cierta información sensible para que al llegar a la configuración del Apimocker pueda encontrar la interacción del recurso con los datos en claro y nos muestre el mock definido como respuesta a la solicitud.
2. El llamado del Shared Flow **sf-cifrado-acceso** en el **PostFlow-Response** cifra la información del payload de respuesta que se considere sensible y que se haya indicado en el **KVM seguridad-campos-cifrar-response** dentro del Configurador de Ambiente (de cada área) para que al llegar a la validación de las respuestas exitosas del feature pueda descifrar ésa información con la función de Gherkin para dicha acción y comparar sus valor en claro con el regex que se hayan configurado para dicha validación.

KVM seguridad-campos-cifrar-response

Para crear las entradas del KVM **seguridad-campos-cifrar-response** en el Configurador de Ambiente se hace de la siguiente manera:

```
{
                    "name":"seguridad-campos-cifrar-response",
                    "entry":[{
                            "name":"capital-humano-desarrollo-personal-informacion-empleados-v1-RHVE-193",
                            "value":"{\"GET/*/beneficiarios\":\"$.resultado.beneficiarios.principales[*].nombre, $
.resultado.beneficiarios.principales[*].apellidoPaterno, $.resultado.beneficiarios.principales[*].apellidoMaterno,
 $.resultado.beneficiarios.opcionales[*].nombre, $.resultado.beneficiarios.opcionales[*].apellidoPaterno, $.result
ado.beneficiarios.opcionales[*].apellidoMaterno, $.resultado.beneficiarios.secundarios[*].nombre, $.resultado.bene
ficiarios.secundarios[*].apellidoPaterno, $.resultado.beneficiarios.secundarios[*].apellidoMaterno\"}"
                        }
                    ]
                }
```

En éste caso la configuración se agregó en el archivo **edge.json** ya que para el ejemplo estamos probando la funcionalidad de cifrado con features a nivel **rama** del Api de ejemplo, si lo desean probar en master u otro ambiente lógico deben agregar la configuración en su archivo correspondiente.